A preliminary version is presented in Asiacrypt 2009.

A Framework for Universally Composable Non-Committing Blind Signatures

Masayuki Abe Miyako Ohkubo

Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
{abe.masayuki,ookubo.miyako}@lab.ntt.co.jp

Abstract

A universally composable (UC) blind signature functionality requres users to commit to the message to be blindly signed. It is thereby impossible to realize in the plain model. This paper shows that even non-committing variants of UC blind signature functionality can not be realized in the plain model. We characterize UC non-committing blind signatures in the common reference string model by presenting equivalent stand-alone security notions under static corruption. Usefulness of the characterization is demonstrated by showing that Fischlin's basic stand-alone blind signature scheme can be transformed into a UC non-committing blind signature protocol without using extra cryptographic components. We extend the results to the adaptive corruption model and present analogous notions, theorems, and constructions both in the erasure model and the non-erasure model.

Keywords: Blind Signatures, Non-committing Blind Signatures, Universal Composability

Contents

1 Introduction

Background. Since the introduction of blind signatures [15], vast numbers of papers have been devoted to efficient constructions, security analysis, and functional extensions, e.g., [30, 24, 33, 3, 4, 26, 7, 17, 29, 5, 8, 1, 2, 34, 25]. Major applications of blind signatures include untraceable payment systems [15] and anonymous electronic voting [16, 19]. The standard notions of security for blind signature schemes in a stand-alone setting are *blindness* and *(one-more) unforgeability* [15, 30, 24]. Canetti's universal composability (UC) framework [9] offers security in a more general setting where other arbitrary protocols are running concurrently. It ensures that the properties guaranteed by an idealized functionality are retained even under general composition. A blind signature functionality was first suggested by Canetti in [10] and formally defined by Fischlin in [17]. A round-optimal realization in the common reference string (CRS) model is presented in [17]. Adaptive security is addressed by Kiayias and Zhou in [26].

In known blind signature functionalities, e.g., [17, 26], a user *commits* to a message to request a signature. Then a signature is issued *remotely* by the functionality from the view of the signer. In [17], 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 is impossible to realize in the plain model [13]. A more formal argument is given by Lindell in [27, 28]. A common idea for these arguments is that the existence of a simulator implies extraction of the input message and hence contradicts the blindness.

Is there a hope to circumvent the impossibility if the functionality is relaxed by sacrificing the commitment property? In some applications such as a simple e-cash or coupon system, every message can be a random string that the users do not need to know or fix in advance. Blindness and unforgeability are the only concerns. In [6], Buan, Gjøsteen, and Kråkmo present 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 is shown that such a non-committing blind signature functionality is realizable in the plain model and that the presented security is equivalent to the unforgeability and weak blindness defined by Juels, Luby and Ostrovsky in [24].

Our contribution. Somewhat in contradiction, we begin by showing a negative result: 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 the unforgeability in the plain model. Importantly, the positive results in [6] holds only for 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 so strong that it is equivalent to incorporating a trusted party in the protocol. While their results show interesting trade-off between the feasibility in the plain model and the restriction on the corruption model, the model poses a serious limitation. Our negative result holds for the most general corruption model.

Despite the negative result, non-committing blind signatures remain an interesting subject to study. The less demanding functionality allows simple protocol designs at least in the conceptual sense. This paper presents a thorough characterization of a non-committing blind signature functionality in the static corruption model. We provide a pair of stand-alone security notions in the CRS model that capture the same security properties as the functionality does. One of the notions is standard unforgeability. The other is a new strong notion of blindness that we call *simulation blindness* (see Section 3). Simulation blindness is, however, an intricate notion that may not necessarily help reduce the burden of proving security or inspire new constructions. We therefore break it up into more easily applicable notions in a special but reasonable setting.

The resulting notions are called *session simulatability* and *signature simulatability*. They literally assert blindness through separate simulations for protocol execution and the resulting signatures.

To demonstrate the usefulness of our characterization, we show that Fischlin's basic blind signature scheme [17] can be transformed into a UC non-committing blind signature protocol without the addition of extra cryptographic components. It is a surprise that such a conceptually simple scheme can be universally composable.

The above results are then extended to the adaptive corruption model. We present notions and theorems that are analogous to those in the static case. We show that, in the erasure model, Fischlin's basic scheme can be adaptively UC secure only by incorporation of a trapdoor into the commitment scheme. Without secure erasures, we construct a scheme based on another variant of Fischlin's scheme that retains its conceptual simplicity. The simplicity of these results can be highlighted when compared to the results of adaptive security for blind signatures of the committing type [26].

2 Preliminaries

2.1 Notations

All algorithms appearing in this paper are probabilistic and 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 randomness r chosen uniformly from an appropriate domain, and the output is something labeled as y. If the randomness is not of concern, we write $y \leftarrow A(x)$ for short. By $(a,b) \leftarrow \langle A(x), B(y)\rangle$ we denote an execution of interactive Turing machines A and B on inputs x and y and with outputs 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. When algorithm A has extra output ω that may be used or ignored depending on the context, we write $a[\omega] \leftarrow A$. The meaning of the symbol in the brackets will be noted whenever this notation is used. An empty string is denoted by $\emptyset$.

2.2 Universal Composability Framework

This section gives a very brief overview of Canetti's UC framework [9]. For details and formal descriptions, please refer to [12].

In the UC framework, a protocol π is compared with an ideal functionality $\mathcal{F}$ considered as a trusted party for the specific tasks. Every player can send a command to $\mathcal{F}$ in a secure and authentic manner, and $\mathcal{F}$ will faithfully carry out the command according to its specification. Such a computation model is called the ideal model. The security of a protocol π for realizing $\mathcal{F}$ is defined according to the simulation paradigm. That is, a protocol π is regarded as secure if, for any adversary $\mathcal{A}$ attacking protocol π, there exists a simulator $\mathcal{S}$ attacking ideal functionality $\mathcal{F}$, whose output distribution is indistinguishable from that of $\mathcal{A}$ for any distinguisher $\mathcal{Z}$ called an environment. Slightly more formally, let random variable $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$ denote the output of $\mathcal{Z}$ with input $a \in \{0,1\}^*$ after observing an execution of π with security parameter $\lambda \in \mathbb{N}$ and uniformly chosen randomness for every player and $\mathcal{A}$. Let $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\lambda, a)$ denote the output of $\mathcal{Z}$ regarding the ideal-model computation. The protocol π securely realizes $\mathcal{F}$ if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that, for any environment $\mathcal{Z}$ and for all $a \in \{0,1\}^*$, $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$ and $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\lambda, a)$ are indistinguishable in λ. If players running protocol π are allowed to access a functionality $\mathcal{F}'$, it is called the $\mathcal{F}'$-hybrid model. Then the output of $\mathcal{Z}$ in the $\mathcal{F}'$-hybrid model is denoted by $\text{EXEC}^{\mathcal{F}'}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$. Adversary $\mathcal{A}$ can corrupt players, and

the corrupted players are completely controlled by $\mathcal{A}$. In the *static* corruption model, $\mathcal{A}$ can corrupt the players only at the first activation. In the *adaptive* model, $\mathcal{A}$ is allowed to corrupt the players at arbitrary times.

A remarkable property of the framework is that it provides a general composition theorem. Roughly, for any protocol ρ that securely realizes functionality $\mathcal{G}$ by using $\mathcal{F}$ as a subroutine, composed protocol ρ^{π} that replaces $\mathcal{F}$ with a secure protocol π also securely realizes $\mathcal{G}$. Intuitively, the composition theorem guarantees that the security properties described in $\mathcal{F}$ are retained even if the corresponding protocol π is used with arbitrary protocols.

3 Blind Signature Scheme

3.1 Syntax and Standard Security Notions

Definition 1 (Blind Signature Scheme). A blind signature scheme BS in the common reference string model consists of five algorithms $\mathsf{BS} = \mathsf{BS}.\{\mathsf{Crs}, \mathsf{Key}, \mathsf{User}, \mathsf{Signer}, \mathsf{Vrf}\}$ as follows.

$\Sigma \leftarrow \mathsf{BS.Crs}(1^{\lambda})$: A common reference string generator that takes as input a security parameter λ and outputs a common reference string Σ.

$(sk, vk) \leftarrow \mathsf{BS.Key}(\Sigma)$: A key generation algorithm that generates a signing key sk and a verification key vk. The message space $\mathcal{M}$ is associated with vk.

$(\sigma, st) \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m), \mathsf{BS.Signer}(\Sigma, sk) \rangle$. A signature generation protocol that is a pair of probabilistic interactive algorithms such that $\mathsf{BS.User}$ outputs signature σ and $\mathsf{BS.Signer}$ outputs status $st \in \{\mathsf{abort}, \mathsf{completed}\}$.

$0/1 \leftarrow \mathsf{BS.Vrf}(\Sigma, vk, \sigma, m)$. A verification algorithm that outputs 1 or 0 for "accept" or "reject," respectively.

A blind signature scheme must provide *completeness* and *consistency* as well as ordinary digital signature schemes. 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). In most general cases, these notions accept a negligible amount of errors. We refer to [20] for details and discussions of these properties.

Definition 2 (Unforgeability: UF). Let $\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) = \Pr[\mathbf{Forge}^{\mathsf{BS}}_{F^*}(\lambda) = 1]$ where $\mathbf{Forge}^{\mathsf{BS}}_{F^*}$ is the experiment shown below. A blind signature scheme BS is unforgeable if $\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda)$ is negligible in λ for any algorithm F^*.

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,
    $m_i \neq m_j$ for all $1 \leq i < j \leq k+1$, and
    $\mathsf{BS.Vrf}(\Sigma, vk, m_i, \sigma_i) = 1$ for all $1 \leq i \leq k+1$.

F^* is allowed to concurrently access the oracle an arbitrary number of times.

By requiring $(m_i, \sigma_i) \neq (m_j, \sigma_j)$ instead of $m_i \neq m_j$, we have the notion of strong unforgeability (sUF). This paper focuses on the above relatively weaker notion because it suffices for major applications.

Definition 3 (Blindness: BL). Let $\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda) = |\Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 0) = 1] - \Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 1) = 1]|$ where $\mathbf{Blind}^{\mathsf{BS}}_{B^*}$ is the experiment shown below. A blind signature scheme BS is blind if $\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda)$ is negligible in λ for any algorithm B^*.

$\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, b)$:
$\Sigma \leftarrow \mathsf{BS.Crs}(1^\lambda)$
$(vk, m_0, m_1) \leftarrow B^*(\Sigma)$
$\sigma_b \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m_b), B^* \rangle_L$
$\sigma_{1-b} \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m_{1-b}), B^* \rangle_L$
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 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 an asterisk (*) throughout this paper.

As observed in [23], the above definition captures the case where the adversary attempts to obtain useful information by aborting the sessions. [18] extends the notion in such a way that, when adversary B^* is given $(\perp, \perp)$ at the end, it is given extra information that determines which session (the first or second or both) actually yields $\perp$ at the user side. The results in this paper also apply to this stronger notion of blindness, called *blindness with selective failure*.

3.2 Blindness based on Simulatability

Blindness can be understood in such a way that any information obtained while executing the signature generation protocol is not included in the resulting signature in a useful form. The following new notion called *simulation blindness* ensures that the signature generation protocol can be executed without involving the message. At the same time, the resulting signature can be generated without involving any information from the protocol run.

Definition 4 (Simulation Blindness: SimBL). A blind signature scheme BS is simulation blind if there exists a tuple of probabilistic algorithms SIM.Crs, SIM.User, and SIM.Sig such that $\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda) = |\Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0) = 1] - \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm D^*, where experiment $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$ is as shown below.

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

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

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

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

D^* is allowed to concurrently access the oracle an arbitrary number of times with arbitrary input messages. Algorithm SIM.User can be stateful but SIM.Sig must be stateless.

The reason we have to handle SIM.User and SIM.Sig in one notion is that, if their properties are stated separately and then the functions are used with the same trapdoor, they may have a negative influence on each other. We thus consider a special case where independent trapdoors are available for each algorithm. For instance, trapdoor t_1 will be used for simulating the signatures and t_2 is for simulating the sessions. We show that, in such a case, the simulation blindness can be deconstructed into more easily applicable notions, which we call *session simulatability* and *signature simulatability*.

Definition 5 (Separable Trapdoor Generator). A probabilistic polynomial-time algorithm SIM.Crs is a separable trapdoor generator if, given security parameter λ, it outputs $(\Sigma, (t_1, t_2))$ such that Σ is indistinguishable from those generated by BS.Crs (with negligible advantage $\mathbf{Adv}^{\mathsf{crs}}_{C^*}$ for any algorithm C^*).

Notice that the separation into two parts is cosmetic in the above definition and the only security requirement is indistinguishability of Σ. More security properties will be demanded in the succeeding security notions. We first introduce signature simulatability where oracle $\mathcal{O}$ plays the role of a user and returns a proper or simulated signature. Adversary A^* then guesses which is the case.

Definition 6 (Signature Simulatability: SigSim). A blind signature scheme BS is signature simulatable with respect to SIM.Crs if there exists a probabilistic algorithm SIM.Sig such that $\mathbf{Adv}^{\mathsf{sis}}_{A^*}(\lambda) = |\Pr[\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 0) = 1] - \Pr[\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm A^*, where experiment $\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, b)$ is as follows.

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

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

$\mathcal{O}_0(\Sigma, vk, m, t_1)$
$\sigma \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m), A^* \rangle_L$
$\sigma' \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t_1)$
If $\sigma = \bot$, set $\sigma' \leftarrow \bot$.
Output σ'

In the next notion of session simulatability, adversary E^* tries to determine whether or not the user is following the proper signature generation protocol with the selected message.

Definition 7 (Session Simulatability: SesSim). A blind signature scheme BS is session simulatable with respect to SIM.Crs if there exists a probabilistic algorithm SIM.User such that $\mathbf{Adv}^{\mathsf{ses}}_{E^*}(\lambda) = |\Pr[\mathbf{SesSIM}^{\mathsf{BS}}_{E^*}(\lambda, 0) = 1] - \Pr[\mathbf{SesSIM}^{\mathsf{BS}}_{E^*}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm E^*, where experiment $\mathbf{SesSIM}^{\mathsf{BS}}_{E^*}$ is as follows.

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

Oracle $\mathcal{O}_1(\Sigma, vk, m, t_2)$:
$\langle \mathsf{BS.User}(\Sigma, vk, m), E^* \rangle$

Oracle $\mathcal{O}_0(\Sigma, vk, m, t_2)$:
$\langle \mathsf{SIM.User}(\Sigma, vk, t_2), E^* \rangle$

Note that the oracles show nothing but the transcript to E^*. In particular, the resulting signatures are never returned. Also note that trapdoor t_1 is given to E^*.

3.3 Relations among Notions

We first show that simulation blindness implies classical blindness. It also implies selective failure blindness. A proof is given in Appendix A.1.

Lemma 1 ($\mathsf{SimBL} \Rightarrow \mathsf{BL}$). *If BS is simulation blind, then it is blind.*

Regarding the reverse direction, we do not know if blindness solely implies simulation blindness or not. However, we 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, 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.

Lemma 2 ($\mathsf{BL} \wedge \mathsf{UF} \not\Rightarrow \mathsf{SimBL}$). *There exists BS that is blind and unforgeable but not simulation blind.*

A proof of Lemma 2 is in Appendix A.2, which helps to understand the impossibility result in Section 4.2.

The next lemma states that it suffices to *separately* consider simulatability of sessions and signatures when separate trapdoors are available for each purpose.

Lemma 3 (SesSim $\wedge$ SigSim $\Rightarrow$ SimBL). *If BS has a separable trapdoor generator and is signature simulatable and session simulatable with respect to the generator, then BS is simulation blind.*

Proof. The proof follows the game transformation paradigm. Starting from $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$, we move to $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$ via three transformations and show that each step changes the view of adversary D^* only in a negligible manner if the underlying scheme BS is session simulatable and signature simulatable. Let X_i denote the event that D^* outputs 1 in Game i.

Game 0. This game is the same as $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$. We thus have

$$\Pr[X_0] = \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)] \tag{1}$$

by definition.

Game 1. Replace $\Sigma \leftarrow \mathsf{BS.Crs}(1^\lambda)$ with $(\Sigma, (t_1, t_2)) \leftarrow \mathsf{SIM.Crs}(1^\lambda)$.

Since SIM.Crs is a separable trapdoor generator, we have

$$\Pr[X_0] - \Pr[X_1] \leq \mathbf{Adv}^{\mathsf{crs}}_{C^*}(\lambda). \tag{2}$$

Game 2. Modify oracle $\mathcal{O}_1$ in such a way that it takes trapdoor t_1 as extra input and computes $\sigma' \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t_1)$ and returns σ' if $\sigma \neq \perp$.

The view of D^* in Game 2 is now the same as that of A^* in $\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 0)$. Since Game 1 is the same as $\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 1)$ with $A^* = D^*$, we have

$$\Pr[X_1] - \Pr[X_2] = \Pr[\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 1)] - \Pr[\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, 0)] \leq \mathbf{Adv}^{\mathsf{sis}}_{A^*}(\lambda). \tag{3}$$

Game 3. Replace $\mathsf{BS.User}(\Sigma, vk, m)$ in Game 2 with $\mathsf{SIM.User}(\Sigma, vk, t_2)$.

We claim that the modification does not change the output distribution of D^* if BS is session simulatable. Formally, there exists E^* such that

$$\Pr[X_2] - \Pr[X_3] \leq \mathbf{Adv}^{\mathsf{ses}}_{E^*}(\lambda). \tag{4}$$

The claim is proven by constructing E^* by using D^* in a straightforward way. (Note that E^* is given trapdoor t_1 and can perform $\sigma \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t_1)$, which is needed to run D^* properly.)

Now observe that Game 3 is identical to $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$. Hence

$$\Pr[X_3] = \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)]. \tag{5}$$

By accumulating (1) to (5), we have

$$\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda) \leq \mathbf{Adv}^{\mathsf{crs}}_{C^*}(\lambda) + \mathbf{Adv}^{\mathsf{sis}}_{A^*}(\lambda) + \mathbf{Adv}^{\mathsf{ses}}_{E^*}(\lambda). \tag{6}$$

∎

4 Universally Composable Non-Committing Blind Signatures

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

Our non-committing blind signature functionality $\mathcal{F}_{\mathrm{ncb}}$ is shown in Figure 1. In the figure, v is a *deterministic* signature verification algorithm. Π is a description of a stateless signing algorithm. See [12] for remarks on running arbitrary algorithms in a functionality. As well as the ordinary signature functionality in [11], 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 key management below.

The idea of using counters to enforce unforgeability is the same as that in [6]. Due to the difference of the timing at which the counters are increased, our formulation can be used with the general communication model thoroughly controlled by the adversary while the one in [6] needs authenticated communication for its realization. Note that the bare signature functionality in [11] can be realized without an authenticated channel because there is no link between the public-key and the identity of the signer and it does not matter who issues a signature as long as the signature is valid.

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

   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.

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

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

to extract the message from the corrupt users.

UNFORGEABILITY. This property holds only while signer P_s is honest. The mechanism to ensure unforgeability is simple. Counter C_{cmpl} counts the number of completed signature generations on 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 this 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 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 corrupt 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 as we assumed v is deterministic. Limiting v to be deterministic decreases generality but makes the exposition considerably simpler. For issues with respect to probabilistic verification algorithms, see [11, 12, 20].

BLINDNESS. Blindness is incorporated by asking $\mathcal{S}$ to register a signature generation function, Π, and using it remotely from the view of $\mathcal{S}$. Some important observations are that Π is fixed *before* any sub-session for signature generation starts, Π takes nothing but message m as input, and message m and $\Pi(m)$ are never sent to $\mathcal{S}$ or P_s during the signature generation phase. This formulation thereby ensures that σ computed remotely by $\sigma \leftarrow \Pi(m)$ is independent of the signer's view in the sub-session for signature generation. Such a paradigm, which we call *remote signing*, is suggested in [10] and used by all known blind signature functionalities. The argument is valid for honest users with registered key $v' = v$.

ON KEY MANAGEMENT. In [10, 12], Canetti formulates the "bare" signature functionality in such a way that the functionality registers a single public-key in every session and the security properties are guaranteed only for the registered public-key. The functionality has concise presentation and high modularity. We use his approach to define $\mathcal{F}_{\mathrm{ncb}}$. Namely, if unregistered v' that is not equal to v is given as input to the signature generation or verification phase, $\mathcal{F}_{\mathrm{ncb}}$ behaves exactly as $\mathcal{S}$ intended. Therefore, even though a user is honest, no security is guaranteed in such a case. (Recall that the environment can pass arbitrary v' to honest users.) Accordingly, upper-level protocols that use $\mathcal{F}_{\mathrm{ncb}}$ must be responsible for providing registered v to honest users. (However, the "registered" public-key can be "faithlessly generated" by the corrupted signer in the key generation phase. This is the case when we argue blindness.)

An alternative approach would be to let $\mathcal{F}_{\mathrm{ncb}}$ explicitly reject unregistered v'. This, however, results in incorporating a mechanism for distributing the correct public-key to honest parties *within the blind signature protocol*. For instance, the protocol realizing $\mathcal{F}_{\mathrm{ncb}}$ may be constructed in the $\mathcal{F}_{\mathrm{ca}}$-hybrid model where $\mathcal{F}_{\mathrm{ca}}$ is the certificate-authority functionality [11] 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 [14], we instead chose to retain basic $\mathcal{F}_{\mathrm{ncb}}$ by outsourcing such a mechanism.

In the literature, [17, 6] implicitly follow the same approach as ours on the key management issue. They, however, define their functionality only for the case of receiving the registered public-key as an input to the signature generation phase. This results in simpler presentation, but eventually the details need to be provided. [26] shows more extended functionality that

handles several public-keys under the same session-id and guarantees blindness for every set of signatures issued with the same public-key. However, this approach suffers high complexity in its presentation. Furthermore, handling multiple keys via the (joint) UC composition theorem would be more reasonable in the UC framework.

VARIATIONS. To focus on essential points, $\mathcal{F}_{\text{ncb}}$ in Figure 1 is defined to notify only the end of the signature generation process to the environment. It can be modified so that the environment can give the signer explicit approval or denial for starting the process by adding another round of interactions among $\mathcal{S}$, $\mathcal{F}_{\text{ncb}}$, and P_s. It is also possible to notify the environment about the abnormal termination of the protocol in the same way. These modifications do not affect the results in this paper since they can be incorporated only by modifying the protocol wrapper in Section 4.3 accordingly.

4.2 Impossibility in Plain Model

This section shows that $\mathcal{F}_{\text{ncb}}$ cannot be realized without accessing extra ideal functionalities or obtaining help from incorruptible parties. We consider non-trivial protocols such that honest parties running the protocol with right inputs terminate and output something with noticeable probability. Note also that we consider the most general case where the environment can corrupt any party at any time.

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

Proof. The outline follows. We use $\mathcal{S}$ to extract remote signing function Π (which is virtually equivalent to sk) and use it to break the unforgeability in the real protocol, which could never happen in the ideal model where absolute unforgeability is provided. Using such $\mathcal{S}$ as a subroutine, $\mathcal{Z}$ distinguishes the ideal process and a real protocol execution.

Suppose that there exists a non-trivial protocol π that realizes $\mathcal{F}_{\text{ncb}}$ in the plain model. Since $\mathcal{F}_{\text{ncb}}$ provides completeness and consistency, so does π. Recall that $\mathcal{F}_{\text{ncb}}$ is invoked by receiving $(\texttt{KeyGen}, sid)$ from a signer. It then outputs $(\texttt{Generated}, sid, v)$ to the signer. Protocol π works in the same way since it realizes $\mathcal{F}_{\text{ncb}}$. Let π_{KG} denote such a part of π that receives $(\texttt{KeyGen}, sid)$ as input and ends with $(\texttt{Generated}, sid, v)$ as output.

Consider a particular pair of adversary $\mathcal{A}^*$ and environment $\mathcal{Z}^*$ that behaves in $\text{EXEC}_{\pi,\mathcal{A}^*,\mathcal{Z}^*}$ as follows. $\mathcal{Z}^*$ first asks $\mathcal{A}^*$ to corrupt the signer. $\mathcal{Z}^*$ then runs π_{KG} with input $(\texttt{KeyGen}, sid)$ and obtains $(\texttt{Generated}, sid, v)$. (Here, without loss of generality, we assume that π_{KG} can be run solely by the signer up to the moment $(\texttt{Generated}, sid, v)$ is output. See the discussion after the proof for generalization.) $\mathcal{Z}^*$ then sends $(\texttt{KeyGen}, sid)$ and v to $\mathcal{A}^*$, and $\mathcal{A}^*$ outputs $(\texttt{Generated}, sid, v)$ on behalf of the corrupt signer. $\mathcal{Z}^*$ then asks for a signature on a message m by sending $(\texttt{Request}, sid, ssid, v, m)$ 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 signer's role and completes the protocol by faithfully following π. The user eventually outputs $(\texttt{Received}, sid, ssid, \sigma)$. Finally $\mathcal{Z}^*$ sends $(\texttt{Verify}, sid, ssid, v, m, \sigma)$ to a user and receives $(\texttt{Verified}, sid, ssid, f)$ as a result of verification. Observe that, even though the signer is corrupted, $\mathcal{Z}^*$ simulates an honest signer by faithfully following protocol 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 $\mathcal{F}_{\text{ncb}}$, there exists a simulator $\mathcal{S}^*$ for such $\mathcal{A}^*$ and $\mathcal{Z}^*$. To successfully simulate $\mathcal{A}^*$, $\mathcal{S}^*$ has to send Π to $\mathcal{F}_{\text{ncb}}$ before $\mathcal{Z}^*$ sends $(\texttt{Request}, sid, ssid, v, m)$ 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 $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$ by using the 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 $\mathcal{Z}^*$ and asks $\mathcal{S}^*$ to corrupt the simulated signer. Next it sends (`KeyGen`, *sid*) and v to $\mathcal{S}^*$ and receives (`Generated`, *sid*, v, Π) from $\mathcal{S}^*$ on behalf of $\mathcal{F}_{\text{ncb}}$. 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 final output of $\mathcal{Z}$ is f.

Let us evaluate $\mathcal{Z}$. Suppose that $\mathcal{Z}$ is in $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. $\mathcal{Z}$ simulates $\mathcal{Z}^*$ perfectly for $\mathcal{S}^*$. In particular, v in this case is generated honestly by π exactly as $\mathcal{Z}^*$ does. Therefore, $\mathcal{S}^*$ 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 $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$. In this case, v is generated by $\mathcal{S}$. If v is distinguishable from the one observed in $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, $\mathcal{Z}$ distinguishes $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$ on that basis. If it is indistinguishable, $\mathcal{S}^*$ outputs (`Generated`, *sid*, v, Π). Since no signature generation process is completed in $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$ and $\mathcal{F}_{\text{ncb}}$ provides absolute unforgeability, $\mathcal{F}_{\text{ncb}}$ rejects σ generated by Π. Thus $f = 0$ for this case. Accordingly, $\mathcal{Z}$ distinguishes $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$ with noticeable probability. ∎

Discussion. An essential observation is that, even though simulator $\mathcal{S}$ does not need to extract the messages from the corrupt users, the functionality still demands that $\mathcal{S}$ extract Π. In addition, it is sufficient to violate one of the claimed properties of the functionality, i.e., unforgeability. The situation is very similar to the case of UC commitments [13], where a simulator must be able to extract the committed message from a corrupt committer and the presence of such a simulator contradicts the hiding property of the commitment protocol.

The proof does not hold 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 its control. This allows $\mathcal{Z}$ to simulate $\mathcal{Z}^*$ simply by sending v generated outside of $\mathcal{Z}$ but by following π_{KG}. If π_{KG} involves parties other than the signer, $\mathcal{Z}^*$ corrupts them before they send any message and simulates 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 with which $\mathcal{Z}$ is working. In slightly more detail, $\mathcal{Z}$ consults with the adversary, $\mathcal{A}$ or $\mathcal{S}$, about the behavior of such players. This works since, in $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, uncorrupted players are running π, and in $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$, simulator $\mathcal{S}$ simulates their behavior in an indistinguishable manner.

The proof holds for any blind signature functionalities, regardless of whether they are committing or non-committing, as long as they follow the remote signing paradigm. The only essential property used in the proof is that $\mathcal{F}_{\text{ncb}}$ receives Π before the signature generation phase for the sake of blindness. We also stress that nothing special is assumed for protocol π that realizes $\mathcal{F}_{\text{ncb}}$. Though remote signing would not be the only way to ensure blindness in the design of functionalities, it is a very natural and, indeed, the only way that has appeared in the literature.

We finally note that the idea of remote signing is also used by the (ordinary) digital signature functionality in [12]. It is interesting to see that the functionality is realizable in the plain model without restriction on the corruption. It is possible because, unlike blind signatures, no security property is left to retain once the signer is corrupted. (Note that it should retain consistency, which can be obtained for free if the verification function is deterministic.) Thus the functionality simply follows the adversary if the signer is corrupted.

4.3 Protocol Wrapper Wrap()

By Wrap(), we denote a simple wrapper algorithm that transforms a stand-alone blind signature scheme BS into a UC blind signature protocol Wrap(BS) in a black-box way without using extra cryptographic components.

The protocol Wrap(BS) is shown in Figure 2. For simplicity, we assume that the security parameter λ is agreed and fixed. The protocol is constructed 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 also as shown in Figure 2.

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

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.

Figure 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 $v' \neq v$. Note also that the signer ignores v' given from the user and uses the genuine secret key sk.

5 Static Security

5.1 Main Theorem

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

Proof. ($\Rightarrow$ direction.) Let $\pi = \mathsf{Wrap}(\mathsf{BS})$. By using $\mathcal{A}$ as a black-box, we construct an $\mathcal{S}$ that simulates the entities and their communication in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model as shown in Figure 3. For such $\mathcal{S}$, we show that $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)$ and $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$ are indistinguishable if the underlying scheme BS is unforgeable and simulation blind.

The proof uses the standard game transformation technique. Starting from $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)$, we gradually modify $\mathcal{F}_{\mathrm{ncb}}$ and $\mathcal{S}$ to obtain a view of $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$ and show that the output distribution of $\mathcal{Z}$ is essentially unchanged by the modifications. In the following, we define a sequence of games, Game 0, ..., Game 5. Let X_i denote the event that $\mathcal{Z}$ outputs 1 in Game i.

Simulator $\mathcal{S}$

Given access to $\mathcal{Z}$ and $\mathcal{A}$, simulator $\mathcal{S}$ simulates entities, $\hat{\mathcal{F}}_{\text{crs}}$, $\hat{P}_u$, $\hat{P}_s$, $\hat{P}_v$ and their communication in $\text{EXEC}^{\mathcal{F}_{\text{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$ while interacting with $\mathcal{F}_{\text{ncb}}$ and $\mathcal{Z}$ in $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$ as follows.

[**In** $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}$]

- (Overall Behavior.) $\mathcal{S}$ first runs $\mathcal{A}$ and let it choose players to corrupt. Then corrupt the corresponding players in IDEAL. On receiving anything on behalf of a corrupt player, forward it to $\mathcal{A}$. On receiving anything from $\mathcal{Z}$, forward it to $\mathcal{A}$. If $\mathcal{A}$ outputs anything to $\mathcal{Z}$, forward it to $\mathcal{Z}$.
- (Key Generation.) On receiving $(\texttt{KeyGen}, sid)$ from $\mathcal{F}_{\text{ncb}}$, forward it to $\hat{P}_s$. If $\hat{P}_s$ outputs $(\texttt{Generated}, sid, v)$, take vk out of v and set $\Pi(m) := \mathsf{SIM.Sig}(\Sigma, vk, m, t)$. Then send $(\texttt{Generated}, sid, v)$ and Π to $\mathcal{F}_{\text{ncb}}$.
- (Signature Generation with Honest P_u.) On receiving $(\texttt{Request}, sid, ssid, v)$ from $\mathcal{F}_{\text{ncb}}$, forward it to $\hat{P}_u$. If $\hat{P}_s$ outputs $(\texttt{Signed}, sid, ssid)$, forward it to $\mathcal{F}_{\text{ncb}}$. If $\hat{P}_u$ outputs $(\texttt{Received}, sid, ssid)$, forward it to $\mathcal{F}_{\text{ncb}}$.
- (Signature Generation with Corrupt P_u.) If $\mathcal{A}$ sends off $(\texttt{Request}, sid, ssid, v)$ to $\hat{P}_s$ on behalf of corrupt $\hat{P}_u$, send $(\texttt{Request}, sid, ssid, v, \perp)$ to $\mathcal{F}_{\text{ncb}}$ on behalf of corrupt P_u in the ideal model. On receiving $(\texttt{Request}, sid, ssid, v)$ from $\mathcal{F}_{\text{ncb}}$, invoke $\hat{P}_s$ by sending $(\texttt{Request}, sid, ssid, v)$ and let it communicate with $\mathcal{A}$. If $\hat{P}_s$ outputs $(\texttt{Signed}, sid, ssid)$, forward it to $\mathcal{F}_{\text{ncb}}$.
- (Signature Verification.) On receiving $(\texttt{Verify}, sid, ssid, v, m, \sigma)$ from $\mathcal{F}_{\text{ncb}}$, forward it to $\hat{P}_v$. If $\hat{P}_v$ outputs $(\texttt{Verified}, sid, ssid, f)$, forward it to $\mathcal{F}_{\text{ncb}}$.

[**In Simulated** $\text{EXEC}^{\mathcal{F}_{\text{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$]

- (Simulating CRS functionality $\hat{\mathcal{F}}_{\text{crs}}$.) On receiving the first request, compute $(\Sigma, t) \leftarrow \mathsf{SIM.Crs}(1^\lambda)$ and returns Σ. Return Σ for further requests.
- (Simulating Honest $\hat{P}_s$ and $\hat{P}_v$) Simply follow the proper protocol as described in Figure 2. If corrupted, follow the instruction of $\mathcal{A}$.
- (Simulating Signature Generation with Honest $\hat{P}_u$.) On receiving $(\texttt{Request}, sid, ssid, v)$, get Σ from $\hat{\mathcal{F}}_{\text{crs}}$, and send $(\texttt{Request}, sid, ssid, v)$ to $\hat{P}_s$. Then invoke $\mathsf{SIM.User}(\Sigma, vk, t)$ and interact with $\hat{P}_s$. If $\mathsf{SIM.User}$ outputs 1, output $(\texttt{Received}, sid, ssid)$.
- (Simulating Communication.) If a simulated entity sends something to another simulated entity, send it it to $\mathcal{A}$ (and let $\mathcal{A}$ decide what to do next). If however the sender or the receiver is $\hat{\mathcal{F}}_{\text{crs}}$ or $\mathcal{A}$, give it directly to the intended receiver.

Figure 3: Simulator $\mathcal{S}$ for the static case.

Game 0. This is the same as $\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)$. Hence

$$\Pr[X_0] = \Pr[\text{IDEAL}_{\mathcal{F}_{\text{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a) = 1]. \tag{7}$$

Game 1. We modify $\mathcal{S}$ and $\mathcal{F}_{\text{ncb}}$ to remove the simulation functions $\mathsf{SIM.Crs}$, $\mathsf{SIM.User}$, and $\mathsf{SIM.Sig}$ as follows.

- In the key generation phase, $\mathcal{S}$ does not send Π to $\mathcal{F}_{\text{ncb}}$. (This removes $\mathsf{SIM.Sig}$.)
- In the signature generation phase, $\mathcal{F}_{\text{ncb}}$ sends m to $\mathcal{S}$. Then $\mathcal{S}$ simulates honest $\hat{P}_u$ by running proper protocol $\mathsf{BS.User}(\Sigma, vk, m)$. If the protocol outputs σ, simulator $\mathcal{S}$ sends it to $\mathcal{F}_{\text{ncb}}$. Then $\mathcal{F}_{\text{ncb}}$ uses this σ in step (ii) of Figure 1 instead of invoking Π. (This removes $\mathsf{SIM.User}$.)
- $\mathcal{S}$ runs proper $\mathsf{BS.Crs}$ in simulating $\hat{\mathcal{F}}_{\text{crs}}$. (This removes $\mathsf{SIM.Crs}$.)

Observe that the view of $\mathcal{Z}$ changes from one that includes simulated signatures and transcripts to one that includes the real signatures and transcripts. The difference, however, should be negligible due to the simulation blindness property. Formally, we claim the following.

Claim 1. *There exists D^* such that, for any $\mathcal{A}$ and $\mathcal{Z}$,* $\Pr[X_0] - \Pr[X_1] \leq \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$.

Proof. By using $\mathcal{Z}$ and $\mathcal{A}$ as black-boxes, we construct D^* that plays in $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$ as follows.

- (Initial Corruption.) Given Σ from $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}$, D^* first runs $\mathcal{Z}$ (and $\mathcal{A}$) and receives the choice of corrupt players. Then D^* simulates $\mathcal{S}$ and $\mathcal{F}_{\mathrm{ncb}}$ as follows.
- (Key Generation.) If $\mathcal{Z}$ sends $(\texttt{KeyGen}, \mathit{sid})$ to honest $\hat{P}_s$, compute $(vk, sk) \leftarrow \mathsf{BS.Key}(1^\lambda)$ and define v properly. Then return $(\texttt{Generated}, \mathit{sid}, v)$ to $\mathcal{Z}$. If $\hat{P}_s$ is corrupted, wait for v to appear in the succeeding commands given from $\mathcal{Z}$. Once it is received, retrieve vk. Either way, output vk to experiment $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}$.
- (Signature Generation with Honest User.) If $\mathcal{Z}$ sends $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v, m)$ to honest $\hat{P}_u$, contact $\mathcal{O}_b$ with m as input. Then if $\hat{P}_s$ is honest, interact with $\mathcal{O}_b$ by running $\hat{P}_s$ with sk. If $\hat{P}_s$ outputs $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$, execute step (i) of $\mathcal{F}_{\mathrm{ncb}}$ and forward $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$ to $\mathcal{Z}$. If $\mathcal{O}_b$ outputs $\sigma \neq \perp$, execute step (ii) of $\mathcal{F}_{\mathrm{ncb}}$ by using the received σ and send $(\texttt{Received}, \mathit{sid}, \mathit{ssid}, \sigma)$ to $\mathcal{Z}$. On the other hand, if $\hat{P}_s$ is corrupt, let $\mathcal{A}$ interact with $\mathcal{O}_b$ on behalf of $\hat{P}_s$. If $\mathcal{O}_b$ outputs $\sigma \neq \perp$, execute step (ii) of $\mathcal{F}_{\mathrm{ncb}}$ by using this σ and send $(\texttt{Received}, \mathit{sid}, \mathit{ssid}, \sigma)$ to $\mathcal{Z}$.
- (Signature Generation with Corrupt User.) If $\mathcal{A}$, working on behalf of a corrupt user, sends $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v)$ to honest $\hat{P}_s$, faithfully simulate honest $\hat{P}_s$ with sk. If $\hat{P}_s$ outputs $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$, execute step (i) of $\mathcal{F}_{\mathrm{ncb}}$ and forward $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$ to $\mathcal{Z}$.
- (Signature Verification.) If $\mathcal{Z}$ sends $(\texttt{Verify}, \mathit{sid}, \mathit{ssid}, v, m, \sigma)$ to P_v, execute the verification process of $\mathcal{F}_{\mathrm{ncb}}$.

Additionally, D^* allows $\mathcal{Z}$ and $\mathcal{A}$ to communicate directly. If $\mathcal{A}$ contacts $\mathcal{F}_{\mathrm{crs}}$, D^* gives Σ to $\mathcal{A}$. When $\mathcal{Z}$ outputs bit $\tilde{b}$ and stops, D^* outputs $\tilde{b}$. This completes the specification of D^*.

In the above reduction, it is important to note that D^* can see message m sent from $\mathcal{Z}$ to honest $\hat{P}_u$. Thus D^* can invoke $\mathcal{O}_b$ with correct m.

Observe that steps (i) and (ii) and the verification processes are done just as specified. So the above simulation maintains list Γ in the same way as $\mathcal{F}_{\mathrm{ncb}}$ does in Game 0 (and Game 1). Therefore, the view $\mathcal{Z}$ obtains from the verification process is in the proper distribution.

Now suppose that D^* is playing in $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$. The CRS is then the output of $\mathsf{SIM.Crs}$. Since D^* is interacting with $\mathcal{O}_0$, the user-side of the signature generation process involves $\mathsf{SIM.User}$ and the returned signature is created by $\mathsf{SIM.Sig}$. Hence the joint view of $\mathcal{Z}$ and $\mathcal{A}$ is identical to one obtained by interacting with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{ncb}}$ as in Game 0. Next suppose that D^* is playing in $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$. The CRS is the real one created by $\mathsf{BS.Crs}$. Since D^* is interacting with $\mathcal{O}_1$, the signature generation process is run faithfully just like it was run in Game 1. Accordingly, the joint view of $\mathcal{Z}$ and $\mathcal{A}$ is a proper one obtained in Game 1. Since the final decision of D^* follows that of $\mathcal{Z}$, we have

$$\Pr[X_0] - \Pr[X_1] \;=\; \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0) = 1] - \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1) = 1] \leq \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$$

as claimed. ∎

Game 2. Modify $\mathcal{F}_{\text{ncb}}$ so that it sets $f = \varphi$ in step 4(b) of the signature verification process. Observe that this modification makes the counter meaningless since the verification process always follows the result of the verification function in step 4. The difference appears only if $\varphi = 1$ happens when entering step 4(b), and we show that such an event implies breaking of the unforgeability. Note that if P_s is corrupt, step 4(b) is never executed. Hence it is sufficient to consider the case where P_s is not corrupt.

Let EventF denote the event such that $\varphi = 1$ happens when entering step 4(b). The view of $\mathcal{Z}$ changes between Game 1 and Game 2 if and only if EventF happens. It thus holds that $\Pr[X_1] - \Pr[X_2] \leq \Pr[\mathsf{EventF}]$. We claim the following.

Claim 2. *There exists F^* such that, for every $\mathcal{A}$ and $\mathcal{Z}$, $\Pr[X_1] - \Pr[X_2] \leq \Pr[\mathsf{EventF}] \leq \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda)$.*

Proof. We show that, if there exists $\mathcal{Z}$ and $\mathcal{A}$ that cause EventF, then we can construct an adversary F^* that breaks the weak unforgeability of BS. Thus $\Pr[\mathsf{EventF}] \leq \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda)$.

Suppose that EventF happens while processing $(\texttt{Verify}, sid, ssid, v, m^\star, \sigma^\star)$. Let Γ be the list held by $\mathcal{F}_{\text{ncb}}$ at that moment. Let $\Gamma_{honest} \subseteq \Gamma$ be the collection of $(m, \sigma, 1)$ recorded in the signature generation process. Let Γ_{uniq} denote a subset of Γ_{honest} that contains all distinct messages. Also let $\Gamma_{4.a} \subseteq \Gamma$ be a collection of $(m, \sigma, 1)$ that causes $f = 1$ in step 4(a). Let $k_{honest} = |\Gamma_{honest}|$, $k_{uniq} = |\Gamma_{uniq}|$, and $k_{4.a} = |\Gamma_{4.a}|$. Then the following hold.

1. $C_{\mathsf{cmpl}} = C_{\mathsf{valid}}$ (since step 4(a) of the verification process was not executed for this input.)
2. $C_{\mathsf{valid}} = k_{uniq} + k_{4.a}$ (since C_{valid} is increment only if a unique message is signed or a signature is accepted in step 4(a).)
3. P_s is not corrupt and $(m^\star, *, 1) \notin \Gamma$. (Otherwise, step 3 of the verification process of $\mathcal{F}_{\text{ncb}}$ must have been executed.) Hence $(m^\star, *, 1) \notin \Gamma_{uniq}$ nor $(m^\star, *, 1) \notin \Gamma_{4.a}$.
4. For every $(m, \sigma, 1) \in \Gamma_{uniq}$, $(m, *, 1) \notin \Gamma_{4.a}$. (Otherwise, step 3 of the verification process must have been executed.) Thus $\Gamma_{uniq} \cap \Gamma_{4.a} = \emptyset$.

Let Γ_{forge} be $\Gamma_{forge} = \Gamma_{uniq} \cup \Gamma_{4.a} \cup \{(m^\star, \sigma^\star, 1)\}$. From the above facts, it holds that Γ_{forge} contains $k_{uniq} + k_{4.a} + 1 = C_{\mathsf{cmpl}} + 1$ valid signatures on distinct messages. Since the signer completes the signature generation process $k = C_{\mathsf{cmpl}}$ times, Γ_{forge} contains a successful $k + 1$ forgery.

Given the above reasoning, we can construct successful F^* from $\mathcal{A}$ and $\mathcal{Z}$ that causes EventF. That is, F^* runs $\mathcal{A}$ and $\mathcal{Z}$ by simulating $\mathcal{S}$ with the help of the signing oracle $\langle \cdot, \mathsf{BS.Signer}(\Sigma, sk) \rangle$ in $\mathbf{Forge}^{\mathsf{BS}}_{F^*}(\lambda)$. (Remember that the signer is honest here.) If EventF happens, F^* outputs Γ_{forge}, which contains $k + 1$ valid signatures on distinct messages. ∎

The following sequence of games, Game 3,...,Game 5, only has superficial modifications to $\mathcal{F}_{\text{ncb}}$ and $\mathcal{S}$ that do not affect the view of $\mathcal{Z}$.

Game 3. Modify the signature verification process of $\mathcal{F}_{\text{ncb}}$ in such a way that, on receiving $(\texttt{Verify}, sid, ssid, v, m, \sigma)$, it computes $\varphi = v(m, \sigma)$, records (m, σ, φ) to Γ, and sends $(\texttt{Verified}, sid, ssid, \varphi)$ to P_v.

We argue that this modification does not change the view of $\mathcal{Z}$, i.e., $\Pr[X_2] = \Pr[X_3]$. Since Game 3 simply replaces f with φ in its output to P_v, the difference appears only if there exists a case that $f \neq \varphi$ in Game 2. In Game 2, the value of f in steps 2 and 3 of the verification process follows that of φ. In step 1, f follows f'. There, entry $(m, \sigma, [f'])$ could have been recorded in the signature generation process that generates σ or in the past verification process that verified (m, σ). If $(m, \sigma, 1)$ has been recorded in the signature generation process, it has been verified once by the honest user so that $\mathsf{BS.Vrf}(\Sigma, vk, \sigma, m) = 1$ holds. Hence due to the completeness (or consistency if the signer is corrupt), $f' = 1 = \varphi$. On the other hand, if the record is made in the past signature verification process, f' is set to φ in step 2 or 3 of the past verification process. Accordingly, in either case, $f' = \varphi$ holds.

Game 4. Modify the signature generation process of $\mathcal{F}_{\mathrm{ncb}}$ by removing the halting condition. (Thus $\mathcal{F}_{\mathrm{ncb}}$ no longer checks $(m, \sigma, 0) \in \Gamma$ in the signing process.)

The halting condition could hold only if $\mathsf{BS.Vrf}(\Sigma, vk, \sigma, m) = 0$ was observed in the verification process, but later $\mathsf{BS.Vrf}(\Sigma, vk, \sigma, m) = 1$ happens as a result of a completed signature generation process. Due to the completeness (or the consistency in the case of a corrupt signer) of the verification protocol, such an event could never happen in Game 3. Thus removing the condition in Game 4 does not change the view of $\mathcal{Z}$ and $\Pr[X_3] = \Pr[X_4]$ stands.

Game 5. Modify the signature generation process of $\mathcal{F}_{\mathrm{ncb}}$ in such a way that it no longer handles Γ and the counters. On receiving $(\texttt{Signed}, sid, ssid, \sigma)$ from $\mathcal{S}$ in the signature generation process, $\mathcal{F}_{\mathrm{ncb}}$ simply forwards it to P_u.

Observe that in Game 4, Γ, C_{valid}, and C_{cmpl} are referred nowhere. Hence, the modification for this game simply removes relevant useless actions in $\mathcal{F}_{\mathrm{ncb}}$. Thus, it does not change the view of $\mathcal{Z}$ at all and $\Pr[X_4] = \Pr[X_5]$ stands.

In Game 5, $\mathcal{F}_{\mathrm{ncb}}$ does nothing but connect the dummy players and $\mathcal{S}$. Also $\mathcal{S}$ does nothing but honestly simulate $\mathcal{F}_{\mathrm{crs}}$, P_u, and P_s that execute $\mathsf{Wrap}(\mathsf{BS})$ in the presence of $\mathcal{A}$ by forwarding the inputs from $\mathcal{F}_{\mathrm{ncb}}$ to the right entities as input from $\mathcal{Z}$, and by returning their outputs (headed to $\mathcal{Z}$) to $\mathcal{F}_{\mathrm{ncb}}$. Therefore, the view of $\mathcal{Z}$ in Game 5 is that obtained by executing protocol $\mathsf{Wrap}(\mathsf{BS})$ in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model. Thus we have

$$\Pr[X_5] = \Pr[\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)]. \tag{8}$$

From the bounds in (7) and (8) and Claim 1 and 2, we have

$$\Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)] - \Pr[\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)] \leq \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda) + \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda), \tag{9}$$

which is negligible due to the unforgeability and simulation blindness of BS.

($\Leftarrow$ direction.) We start by showing that, if BS is not simulation blind, then for any $\mathcal{S}$ there exists $\mathcal{Z}$ that successfully distinguishes IDEAL and EXEC. It is done in two steps. First we construct simulation algorithms $\mathsf{SIM.Crs}$, $\mathsf{SIM.Sig}$, and $\mathsf{SIM.User}$ by using $\mathcal{S}$ as a subroutine. Since BS is not simulation blind, there exists D^* that successfully breaks the simulation blindness with respect to those simulation algorithms. We then construct $\mathcal{Z}$ that uses D^* as a subroutine. $\mathcal{Z}$ interacts with the adversary, which is either $\mathcal{S}$ or $\mathcal{A}$, so that their interaction simulates $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$ or $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$, respectively. Accordingly, if D^* distinguishes the simulation algorithms from the real ones, $\mathcal{Z}$ distinguishes $\mathcal{S}$ and $\mathcal{A}$ as well. Note that we take advantage of $\mathcal{Z}$ being specific to $\mathcal{S}$.

A technical difficulty is that algorithms $\mathsf{SIM.Crs}$, $\mathsf{SIM.Sig}$, and $\mathsf{SIM.User}$ are independent functions and have access to an independent copy of $\mathcal{S}$. Since $\mathcal{S}$ is probabilistic, these functions

need to give the same randomness to $\mathcal{S}$ so that $\mathcal{S}$ behaves in exactly the same way for each algorithm. Our idea is to use trapdoor t as a container for the randomness given to $\mathcal{S}$. Details follow.

Consider adversary $\mathcal{A}$ that behaves as follows. It first corrupts the signer P_s and outputs Σ to $\mathcal{Z}$. It then receives $(\texttt{KeyGen}, \mathit{sid})$ and v from $\mathcal{Z}$. On further receiving $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v)$, it asks $\mathcal{Z}$ to take over the role of the signer. It will never block any communication. Let $\mathcal{S}$ be a simulator for such $\mathcal{A}$. We define $\mathsf{SIM.Crs}$, $\mathsf{SIM.Sig}$, and $\mathsf{SIM.User}$ as follows.

- $\mathsf{SIM.Crs}$: On input security parameter λ and randomness r, run $\mathcal{S}$ with randomness r. $\mathcal{S}$ corrupts signer P_s and outputs Σ. Set $t = r$ and output (Σ, t).
- $\mathsf{SIM.Sig}$: On input $(\Sigma, vk, m, t; \xi)$, run $\mathcal{S}$ with randomness $t(= r)$ and do the same as $\mathsf{SIM.Crs}$ does up to the point $\mathcal{S}$ outputs Σ. (This procedure is needed to make $\mathcal{S}$ select the same Σ.) Then properly form v with respect to vk and send $(\texttt{KeyGen}, \mathit{sid})$ and v to $\mathcal{S}$. When $\mathcal{S}$ outputs $(\texttt{Register}, \mathit{sid}, v, \Pi)$, compute $\sigma \leftarrow \Pi(m; \xi)$ and output σ.
- $\mathsf{SIM.User}$: This algorithm is stateful. On input (Σ, vk, t), run $\mathcal{S}$ with randomness $t(= r)$ and do the same as $\mathsf{SIM.Sig}$ up to the moment that $\mathcal{S}$ outputs $(\texttt{Register}, \mathit{sid}, v, \Pi)$. (This procedure is done only for the first invocation of $\mathsf{SIM.User}$.) Then send $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v)$ to $\mathcal{S}$ to let $\mathcal{S}$ start simulating the user-side in the signature generation protocol. Output 1 if $\mathcal{S}$ outputs $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$. Next time $\mathsf{SIM.User}$ is called, send $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v)$ with fresh ssid to $\mathcal{S}$ and repeat the process.

Note that, in $\mathsf{SIM.Sig}$, simulator $\mathcal{S}$ will output $(\texttt{Register}, \mathit{sid}, v, \Pi)$ to complete the key generation process before any signature generation process starts since $\mathcal{A}$ does so in the real protocol. Also note that, in $\mathsf{SIM.User}$, $\mathcal{S}$ must output $(\texttt{Signed}, \mathit{sid}, \mathit{ssid})$ in polynomial time whenever the signer completes the protocol since $\mathcal{A}$ does not interfere with the communication and the corresponding honest user outputs a signature in polynomial time.

Since we assumed that BS is not simulation blind, there exists an algorithm D^* whose advantage $\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$ with respect to the above $(\mathsf{SIM.Crs}, \mathsf{SIM.Sig}, \mathsf{SIM.User})$ is not negligible. We now construct $\mathcal{Z}$ that uses D^* by simulating $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$ as follows.

- $\mathcal{Z}$ runs the adversary, $\mathcal{A}$ or $\mathcal{S}$. The adversary corrupts the signer and sends Σ to $\mathcal{Z}$.
- $\mathcal{Z}$ executes $vk \leftarrow D^*(\Sigma)$ and properly forms v with respect to vk.
- $\mathcal{Z}$ sends $(\texttt{KeyGen}, \mathit{sid})$ and v to the adversary working on behalf of the corrupt signer.
- If D^* accesses $\mathcal{O}_b$ with message m, $\mathcal{Z}$ sends $(\texttt{Request}, \mathit{sid}, \mathit{ssid}, v, m)$ to an honest user to invoke the signature generation process. If the adversary asks $\mathcal{Z}$ to interact with the user on behalf of the corrupt signer, let D^* take over the interaction. (Remember that D^* plays the role of the signer while it interacts with $\mathcal{O}_b$.) When the user outputs $(\texttt{Received}, \mathit{sid}, \mathit{ssid}, \sigma)$, send σ to D^*.

Let us verify the view of D^*. Consider the case that $\mathcal{Z}$ is working with adversary $\mathcal{A}$ in $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a)$. Σ is sampled from $\mathsf{BS.Crs}$, and the signatures given to D^* are generated by the protocol execution between an honest user running $\mathsf{BS.User}$ and D^*. Hence, it corresponds to the outputs from $\mathcal{O}_1$ and the view of D^* is that of $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$. Next consider the case that $\mathcal{Z}$ is working with $\mathcal{S}$ in $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a)$. CRS Σ is that given from $\mathcal{S}$ as specified by $\mathsf{SIM.Crs}$. Observe that the interaction during the signature generation process is simulated by $\mathcal{S}$ as specified by $\mathsf{SIM.User}$. Observe also that the resulting signatures returned from $\mathcal{F}_{\mathrm{ncb}}$ are

created by Π registered by $\mathcal{S}$ exactly as specified by $\mathsf{SIM.Sig}$. The distribution thus corresponds to those from $\mathcal{O}_0$. Consequently, the view of D^* is that of $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$. Accordingly,

$$\begin{aligned}
&|\Pr[\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a) = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a) = 1]| \\
&= |\Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1) = 1] - \Pr[\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0) = 1]| \qquad (10) \\
&= \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda),
\end{aligned}$$

which is not negligible if BS is not simulation blind.

We have shown that if $\mathsf{Wrap}(\mathsf{BS})$ securely realizes $\mathcal{F}_{\mathrm{ncb}}$ then BS is simulation blind. We now show the "and" part of the theorem by constructing successful $\mathcal{Z}$ from adversary F^* that breaks the unforgeability of BS assuming that BS is simulation blind. $\mathcal{Z}$ is as follows.

- $\mathcal{Z}$ runs F^*. In responding to the request from F^*, $\mathcal{Z}$ obtains k correct signatures through an honest user. Eventually F^* outputs $k+1$ signatures on distinct messages.
- If F^* fails to create $k+1$ valid signatures that pass the verification function of BS, then flip a coin and output 1 with probability $1/2$ and stop.
- Otherwise, $\mathcal{Z}$ asks an honest user to verify these $k+1$ signatures. If all of them are accepted, output 1. Otherwise, output 0.

If $\mathcal{Z}$ is working in $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}$, the signatures given to F^* are the genuine ones as required in $\mathbf{Forge}^{\mathsf{BS}}_{F^*}$. Thus, with probability $\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda)$, adversary F^* outputs $k+1$ signatures that pass the verification function of BS. In such a case, all $k+1$ signatures will be accepted with probability 1. Therefore,

$$\Pr[\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a) = 1] = \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) \cdot 1 + (1 - \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda)) \cdot \frac{1}{2}. \qquad (11)$$

If $\mathcal{Z}$ is working in $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}$, the signatures given to F^* are the simulated ones. Since BS is simulation blind, F^* is still successful in creating $k+1$ valid signatures with probability of at least $\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) - \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$. However, $\mathcal{Z}$ observes only k acceptance due to the absolute unforgeability posed by the counter. Therefore,

$$\begin{aligned}
\Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a) = 0] &= (\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) - \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)) \cdot 1 + (1 - (\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) - \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda))) \cdot \frac{1}{2} \\
&= \frac{1}{2}(1 + \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) - \mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)),
\end{aligned} \qquad (12)$$

where D^* is an adversary constructed from $\mathcal{Z}$ and attacks simulation blindness. From (11) and (12) we have

$$\Pr[\mathrm{EXEC}^{\mathcal{F}_{\mathrm{crs}}}_{\pi,\mathcal{A},\mathcal{Z}}(\lambda, a) = 1] - \Pr[\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{ncb}},\mathcal{S},\mathcal{Z}}(\lambda, a) = 1] = \mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) - \frac{1}{2}\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda), \qquad (13)$$

which is not negligible if BS is not unforgeable while it is simulation blind. ∎

5.2 Universal Composability of Fischlin's Generic Scheme

We revisit Fischlin's construction of a stand-alone blind signature scheme that provides blindness and unforgeability in the CRS model. While the original scheme in [17] uses a public-key encryption and a non-interactive zero-knowledge proof, our description treats them in a combined way as a non-interactive zero-knowledge proof of knowledge [32]. It allows a simpler description and more abstract arguments. The building blocks are the following. See Appendix B for details of these components.

- $\mathsf{SIG} = \mathsf{SIG}.\{\mathsf{Key}, \mathsf{Sign}, \mathsf{Vrf}\}$: A signature scheme that is existentially unforgeable against chosen message attacks.
- $\mathsf{BC} = \mathsf{BC}.\{\mathsf{Key}, \mathsf{Com}, \mathsf{Vrf}\}$: A commitment scheme that fulfills binding and hiding properties in any flavor.
- $\mathsf{NIZK} = \mathsf{NIZK}.\{\mathsf{Crs}, \mathsf{Prf}, \mathsf{Vrf}, \mathsf{SimPrf}, \mathsf{Ext}\}$: A multi-theorem non-interactive zero-knowledge proof of knowledge for witness (s, c, z) that satisfies relation

$$\mathsf{BC.Vrf}(\Sigma_{\mathsf{bc}}, c, m, z) = 1 \wedge \mathsf{SIG.Vrf}(vk, c, s) = 1. \tag{14}$$

The CRS generation function $\mathsf{BS.Crs}$ computes $(\Sigma_{\mathsf{zk}}, t_{\mathsf{zk}}) \leftarrow \mathsf{NIZK.Crs}(1^\lambda)$ and $\Sigma_{\mathsf{bc}} \leftarrow \mathsf{BC.Key}(1^\lambda)$ and outputs $\Sigma \leftarrow (\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}})$. $\mathsf{BS.Key}$ is the same as $\mathsf{SIG.Key}$, which outputs vk and sk. The signature generation protocol is shown in Figure 4. Verification function $\mathsf{BS.Vrf}$ takes $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}}), vk, \sigma, m)$ as input and outputs $\varphi \in \{0, 1\}$ such that $\varphi \leftarrow \mathsf{NIZK.Vrf}(\Sigma_{\mathsf{zk}}, (vk, \Sigma_{\mathsf{bc}}, m), \sigma)$.

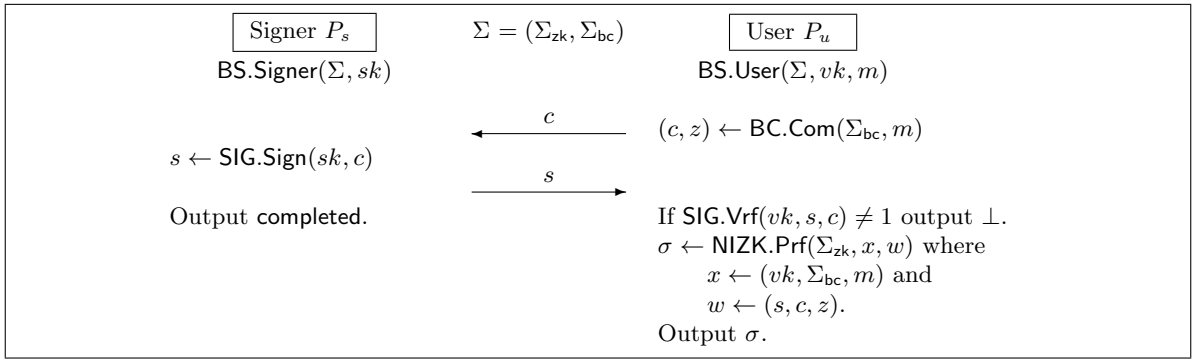

Figure 4: Fischlin's Basic Blind Signature Scheme $\mathsf{BS_G}$. Signature Generation Protocol.

Proposition 1. *Protocol $\mathsf{Wrap}(\mathsf{BS_G})$ securely realizes $\mathcal{F}_{\text{ncb}}$ in the $\mathcal{F}_{\text{crs}}$-hybrid model with respect to static adversaries.*

Proof. According to Theorem 2 and Lemma 3, it suffices to show that $\mathsf{BS_G}$ is unforgeable, signature simulatable, and session simulatable for a separable trapdoor generator.

Claim 3. *$\mathsf{BS_G}$ is unforgeable if NIZK is knowledge sound, SIG is existentially unforgeable against chosen message attacks, and BC is binding.*

If the adversary outputs $k+1$ valid (m, σ), then $\mathsf{NIZK.Ext}$ extracts $k+1$ tuples of (m, s, c, z). Due to the knowledge soundness of NIZK, they must satisfy $\mathsf{BC.Vrf}(\Sigma_{\mathsf{bc}}, c, m, z) = 1$ and $\mathsf{SIG.Vrf}(vk, c, s) = 1$. (Note that simulation soundness is not needed here since we can provide correct witnesses for the reduction.) Now, if there is c that is not included in the signer's

view, the existential unforgeability of SIG is broken. On the other hand, if no such c exists, by the pigeon-hole principle, there must be a pair (m, s, c, z) and (m', s', c, z') where $m \neq m'$. This breaks the binding property of BC.

Claim 4. *BS_G has a separable trapdoor generator.*

We construct $\mathsf{SIM.Crs}$ as follows. Given λ, it computes $(\Sigma_{\mathsf{zk}}, t_{\mathsf{zk}}) \leftarrow \mathsf{NIZK.Crs}(1^\lambda)$ and $\Sigma_{\mathsf{bc}} \leftarrow \mathsf{BC.Key}(1^\lambda)$, and outputs $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}}), (t_{\mathsf{zk}}, \emptyset))$, where t_{zk} is the first part and the second part is empty. Then from the zero-knowledge property of NIZK, $(\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}})$ is indistinguishable from the real ones. The trapdoors are also trivially independent of each other. $\mathsf{SIM.Crs}$ is thus a separable trapdoor generator for BS_G.

Claim 5. *BS_G is signature simulatable with respect to the above $\mathsf{SIM.Crs}$ if NIZK is multi-theorem zero-knowledge.*

Define $\mathsf{SIM.Sig}$ as an algorithm such that, given $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}}), vk, m, t_{\mathsf{zk}})$, it computes $\sigma \leftarrow \mathsf{NIZK.SimPrf}(t_{\mathsf{zk}}, (vk, \Sigma_{\mathsf{bc}}, m))$ and outputs σ. Then it is straightforward to show that breaking the signature simulatability for such $\mathsf{SIM.Sig}$ implies breaking the multi-theorem zero-knowledge property of NIZK.

Claim 6. *BS_G is session blind with respect to the above $\mathsf{SIM.Crs}$ if BC is hiding.*

Define $\mathsf{SIM.User}$ as follows. Given input $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{bc}}), vk, \emptyset)$, $\mathsf{SIM.User}$ selects a random message $m' \in \mathcal{M}$, computes $(c, z) \leftarrow \mathsf{BC.Com}(\Sigma_{\mathsf{bc}}, m')$, and outputs c. On receiving s, it computes $\delta \leftarrow \mathsf{SIG.Vrf}(vk, s, c)$ and outputs δ. Now it is obvious that breaking the session blindness with respect to such $\mathsf{SIM.User}$ implies breaking the hiding property of BC. Note that t_{zk} has to be given to the adversary in the experiment of session blindness. It is, however, independent of BC, and the reduction algorithm can generate it by itself. ∎

5.3 Other Generic Constructions

In the proof of Proposition 1, the signature simulatability of BS_G is shown by simulating the *user*-side algorithm. The observation immediately raises an alternative approach that simulates the *signer*-side algorithm. For this purpose, we introduce a *simulatable signature scheme*. (See Appendix B.6.) It is a signature scheme in the CRS model with an extra function, $\mathsf{SSIG.Sim}$, that takes trapdoor t_{ssig} associated with CRS Σ_{ssig} and generates acceptable signatures as well as the original signing algorithm $\mathsf{SIG.Sign}$. Given such a simulatable signature scheme, we construct $\mathsf{SIM.Sig}$ in such a way that it computes $(c, z) \leftarrow \mathsf{BC.Com}(\Sigma_{\mathsf{bc}}, m)$, $s \leftarrow \mathsf{SSIG.Sim}(\Sigma_{\mathsf{ssig}}, pk_{\mathsf{ssig}}, m, t_{\mathsf{ssig}})$, and $\sigma \leftarrow \mathsf{NIZK.Prf}(\Sigma_{\mathsf{zk}}, x, w)$ for $x = (vk, (\Sigma_{\mathsf{ssig}}, \Sigma_{\mathsf{bc}}), m)$, and $w = (s, c, z)$ and outputs σ. With this structure, the scheme provides signature simulatability based on the witness indistinguishability of NIZK. (Zero-knowledge is no longer needed and witness indistinguishability is sufficient since a correct witness is available.) This particular structure is suggested in [23], where it is used as a starting point for removing the CRS in the stand-alone model. We will take advantage of the structure for achieving adaptive security in Section 6.

The above methodology applies to a class of blind signature schemes that includes classical "blind–sign–unblind" type constructions. Namely, if the "sign" part can be simulatable in the CRS model, the scheme can be turned into a universally composable one by following our framework. This observation suggests revisiting existing efficient schemes in several settings and

seeing if their "sign" part can be efficiently simulatable. In general, as shown in [6, 23], such simulatability at the signer's side is obtained by publishing a NIZK proof of knowledge of the secret key as a part of the public-key.

6 Adaptive Security

6.1 State Reconstructability in Stand-Alone Notions

We first stress that adaptive corruption on the signer can be handled without any difficulty. Observe that, in the proof of Theorem 2, the signer is simulated simply by following the protocol when the signer is not corrupted. However, if the signer is corrupted, simulator $\mathcal{S}$ simply follows $\mathcal{A}$ to simulate the corrupted signer. Accordingly, $\mathcal{S}$ in the adaptive case simulates signer $\hat{P}_s$ faithfully by the moment it is corrupted, then reveals the whole view of $\hat{P}_s$ on corruption and follows $\mathcal{A}$ thereafter. This results in a perfect simulation of the signer.

In contrast, handling adaptive corruption on a user demands *equivocability* through *state reconstruction* so that a consistent view can be provided to $\mathcal{A}$. To capture this strong property, we enhance simulation blindness as follows.

Definition 8 (Equivocal Simulation Blindness: $\mathsf{EqSimBLND}$). A blind signature scheme BS is equivocal simulation blind if there exists a tuple of probabilistic algorithms $\mathsf{SIM.Crs}$, $\mathsf{SIM.User}$, $\mathsf{SIM.Sig}$, and $\mathsf{SIM.State}$ that fulfill the following. $\mathsf{SIM.User}$ and $\mathsf{SIM.State}$ may be stateful and $\mathsf{SIM.Sig}$ is stateless. 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 algorithm D^*, where $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda,b)$ is the same experiment as $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda,b)$ (see Definition 4) with the following oracles.

$\mathcal{O}_1(\Sigma, vk, m, t)$	$\mathcal{O}_0(\Sigma, vk, m, t)$
$\sigma \leftarrow \langle \mathsf{BS.User}(\Sigma, vk, m; r), D^* \rangle_L$	$\delta[\omega_u] \leftarrow \langle \mathsf{SIM.User}(\Sigma, vk, t), D^* \rangle_L$
Output (σ, r)	$\sigma[\omega_s] \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t)$
	$r \leftarrow \mathsf{SIM.State}(\omega_u, \omega_s)$
	If $\delta = 0$, then set $\sigma = \perp$.
	Output (σ, r)

Denoted by ω_u and ω_s are the state information of $\mathsf{SIM.User}$ and $\mathsf{SIM.Sig}$, respectively.

Note that $\mathsf{SIM.State}$ is supposed to simulate randomness even for the case where the interaction between $\mathsf{SIM.User}$ and D^* is terminated abnormally. $\mathsf{SIM.State}$ can see how the interaction is terminated as the view of $\mathsf{SIM.User}$ is given.

Definition 4, 8 and Lemma 1 indicate the following lemma.

Lemma 4. *EqSimBLND* $\Rightarrow$ *BL*

As well as the static case, we provide more easily applicable notions by deconstructing equivocal simulation blindness. In addition to the separable trapdoor generation property, we consider the case where separate randomness is used for blinding the session and the signature.

Definition 9 (Signature Equivocality: SigEq). A blind signature scheme BS is signature equivocal with respect to $\mathsf{SIM.Crs}$ if there exists probabilistic algorithms $\mathsf{SIM.Sig}$ and $\mathsf{SIM.SigState}$ such that $\mathbf{Adv}^{\mathsf{sigeq}}_{A^*}(\lambda) = |\Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda,0) = 1] - \Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda,1) = 1]|$ is negligible in λ for

any algorithm A^*, where experiment $\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, b)$ is the same as $\mathbf{SigSIM}^{\mathsf{BS}}_{A^*}(\lambda, b)$ in Definition 6 except for using modified oracle $\mathcal{O}_b$ as follows.

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

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

Note that the above oracles work exactly the same when $\sigma = \perp$, i.e., BS.User terminates abnormally.

Definition 10 (Session Equivocality: SesEq). A blind signature scheme BS is session equivocal with respect to SIM.Crs if there exists probabilistic algorithms SIM.User and SIM.SesState such that $\mathbf{Adv}^{\mathsf{seseq}}_{E^*}(\lambda) = |\Pr[\mathbf{SesEQ}^{\mathsf{BS}}_{E^*}(\lambda, 0) = 1] - \Pr[\mathbf{SesEQ}^{\mathsf{BS}}_{E^*}(\lambda, 1) = 1]|$ is negligible in λ for any algorithm E^*, where experiment $\mathbf{SesEQ}^{\mathsf{BS}}_{E^*}$ is the same as $\mathbf{SesSIM}^{\mathsf{BS}}_{E^*}$ in Definition 7 except for using the following modified oracles.

$\mathcal{O}_1(\Sigma, vk, m, t_2)$: $\langle \mathsf{BS.User}(\Sigma, vk, m; r_1 \| \| r_2), E^* \rangle$ Return r_2	$\mathcal{O}_0(\Sigma, vk, m, t_2)$: $\delta[\omega_u] \leftarrow \langle \mathsf{SIM.User}(\Sigma, vk, t_2), E^* \rangle_L$ $r_2 \leftarrow \mathsf{SIM.SesState}(\omega_u, m)$ Return r_2

Lemma 5 (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.*

The proof is in Appendix A.3, which is quite similar to the proof of Lemma 3 but additionally considers state reconstruction through SIM.SigState and SIM.SesState.

We next consider restricted simulation blindness for capturing the erasure model. Let EqSimBLND/R be equivocal simulation blindness with a restriction such that the oracles return r only if $\sigma = \perp$ in Definition 8. Namely, randomness is revealed only when the (simulated) signature generation protocol is abnormally terminated. Observe that oracles $\mathcal{O}_1$ and $\mathcal{O}_0$ in Definition 9 are identical when $\sigma = \perp$ and hence SIM.SigState is useless if we only consider such a case. Therefore, if the randomness is returned only in such a case, SIM.SigState can be removed. Since SigEq and SigSim differ only in the presence of SIM.SigState, removing SIM.SigState turns SigEq in Lemma 5 into SigSim. Thus we have the following lemma such that we no longer need equivocability in the signature simulatability to achieve the limited simulation blindness. It will be used in Section 6.3, where the adaptive security of Fischlin's scheme is considered in the erasure model. A proof can be derived from that of Lemma 5.

Lemma 6 (SesEq $\wedge$ SigSim $\Rightarrow$ EqSimBLND/R). *If BS has a separable trapdoor generator and is (regular) signature simulatable and session equivocal with respect to the generator, then BS is equivocal simulation blind with a restriction such that r is returned only if $\sigma = \perp$.*

6.2 Main Theorems in Adaptive Case

Similar to the static case, we present theorems such that unforgeability and equivocal simulation blindness are necessary and sufficient for adaptive UC blind signatures.

Theorem 3 (UF $\wedge$ EqSimBLND $\Leftrightarrow$ $\mathcal{F}_{\mathrm{ncb}}$[adaptive,non-erasure]). *If BS is unforgeable and equivocal simulation blind, then protocol Wrap(BS) securely realizes $\mathcal{F}_{\mathrm{ncb}}$ with respect to adaptive adversaries.*

Proof. ($\Rightarrow$ direction.) The proof is almost the same as that for Theorem 2. We use the simulator shown in Figure 3 with the following modifications to handle adaptive corruptions.

- (Corruption of $\hat{P}_s$). Simulator $\mathcal{S}$ sends $\mathcal{A}$ all the randomness used for simulating $\hat{P}_s$ so far. It includes randomness used for key generation if corruption happens after the key generation process. (This is possible since $\hat{P}_s$ is simulated simply by following the genuine protocol.)

- (Corruption of $\hat{P}_u$). Suppose that $\hat{P}_u$ was once engaged in $\delta[\omega_u] \leftarrow \langle \mathsf{SIM.User}(\Sigma, vk, t_2), \hat{P}_s \rangle_L$. By corrupting P_u in the ideal model, $\mathcal{S}$ obtains (m, σ) observed by $\mathcal{Z}$ with respect to the protocol execution and ξ used as $\sigma \leftarrow \Pi(m; \xi)$ by $\mathcal{F}_{\mathrm{ncb}}$. (In the case of $\delta = 0$, i.e., the protocol is terminated abnormally, only m is given to $\mathcal{S}$. Simulator $\mathcal{S}$ sets ξ randomly in such a case.) $\mathcal{S}$ then constructs ω_s from ξ and Π. (Recall that $\Pi(\cdot)$ is $\mathsf{SIM.Sig}(\Sigma, vk, \cdot, t)$.) It then runs $r \leftarrow \mathsf{SIM.State}(\omega_u, \omega_s)$. $\mathcal{S}$ repeats the above procedure for every invocation of $\mathsf{SIM.User}$ in the simulation of $\hat{P}_u$. (If there is a running copy of $\mathsf{SIM.User}$, it is terminated and δ is set to 0.) $\mathcal{S}$ then sends $\mathcal{A}$ all r obtained as above and all $(\mathtt{Request}, sid, ssid, v, m)$ and $(\mathtt{Received}, sid, ssid, \sigma)$ received from $\mathcal{F}_{\mathrm{ncb}}$.

Once a party is corrupted, $\mathcal{S}$ follows the behavior of $\mathcal{A}$ working on behalf of the corrupted party.

The rest of the proof is the same as that of Theorem 2 except for Game 1. In Game 1 for this case, $\mathcal{S}$ no longer uses $\mathsf{SIM.State}$ in responding corruptions but simply returns the randomness used in the simulation of the corrupted party. (Note that with the modification in Game 1, all parties are simulated faithfully until the moment they are corrupted.) Claim 1 is then restated with upper bound $\mathbf{Adv}^{\mathsf{eqsib}}_{D^*}(\lambda)$.

The reduction algorithm D^* in the proof of Claim 1 is modified as follows.

- Given Σ from $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}$, D^* invokes $\mathcal{Z}$. On receiving $(\mathtt{KeyGen}, sid)$ from $\mathcal{Z}$, if P_s has not been corrupted yet, D^* generates $(vk, sk) \leftarrow \mathsf{BS.Key}(1^\lambda)$ and outputs vk to $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}$. Otherwise, D^* receives $(\mathtt{Generated}, sid, v)$ from $\mathcal{A}$ and outputs vk to $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}$.

- On receiving corruption on $\hat{P}_s$ after key generation, send the view of $\hat{P}_s$ to $\mathcal{A}$.

- If $\mathcal{Z}$ sends $(\mathtt{Request}, sid, ssid, v, m)$ to P_u, contact $\mathcal{O}_b$ with m as input. Then, if P_s is honest, engage in the signature generation protocol by faithfully running P_s with sk. Otherwise, if P_s is corrupt (or corrupted while running the protocol), let $\mathcal{A}$ execute the protocol on behalf of P_s. If P_s outputs $(\mathtt{Signed}, sid, ssid)$, send it to $\mathcal{Z}$. If $\mathcal{O}_b$ outputs (σ, r), send $(\mathtt{Received}, sid, ssid, \sigma)$ to $\mathcal{Z}$ and store r.

- On receiving corruption on user $\hat{P}_u$, do the following. For every invocation of $\mathcal{O}_b$ with respect to $\hat{P}_u$, if (σ, r) has already been returned from the oracle, send r to $\mathcal{A}$. If there is a session that $\mathcal{O}_b$ is still running (i.e., interacting with the signer), suspend the signer and let $\mathcal{O}_b$ terminate abnormally so that it outputs $(\perp, r)$ now. Then return r to $\mathcal{A}$ and resume the signer (so that it can continue the protocol with the corrupt user).

Observe that r received from $\mathcal{O}_b$ are real if $b = 1$, while they are computed by $\mathsf{SIM.State}$ if $b = 0$. The rest of the proof then follows that of Claim 1.

($\Leftarrow$ direction.) The outline of the proof is the same as that of Theorem 2. Construction of $\mathsf{SIM.Sig}$ and $\mathsf{SIM.User}$ is unchanged except that they additionally output the state information ω_s and ω_u, which are the views of the algorithms. To meet the definition of equivocal simulation blindness, we have to construct $\mathsf{SIM.State}$. Consider adversary $\mathcal{A}$ that corrupts a user (on request from $\mathcal{Z}$) and outputs randomness r used by the user. Then simulator $\mathcal{S}$ does the same. Now, on input state information ω_u and ω_s, $\mathsf{SIM.State}$ runs $\mathcal{S}$ up to the point it reaches the same state as $\mathcal{S}$ in $\mathsf{SIM.User}$. Then let $\mathcal{S}$ corrupt the user that made the signature generation request (since the user is a simulated one in $\mathsf{SIM.User}$, its identity is available from ω_u), and give (σ, m, ξ) that corresponds to the target sub-session. Note that randomness ξ is computed from ω_s. Eventually $\mathcal{S}$ outputs r, and $\mathsf{SIM.State}$ outputs r. $\mathcal{Z}$ is then adjusted in the following way. On receiving $(\mathtt{Received}, \mathit{sid}, \mathit{ssid}, \sigma)$ from the user, $\mathcal{Z}$ corrupts the user and receives r from the adversary. It then sends (σ, r) to D^*. The rest of the proof is the same as that for Theorem 2. ∎

In the secure erasure model, we have a similar theorem, as shown in Theorem 4. It can be proven in the same way as that for Theorem 3 with modification so that randomness r is considered only in the case of $\sigma = \perp$.

Theorem 4 ($\mathsf{UF} \wedge \mathbf{EqSimBLND/R} \Leftrightarrow \mathcal{F}_{\mathbf{ncb}}$[adaptive,erasure]). *If BS is unforgeable and equivocal simulation blind with restriction, then protocol $\mathsf{Wrap}(\mathsf{BS})$ securely realizes $\mathcal{F}_{\mathrm{ncb}}$ with respect to adaptive adversaries in the erasure model.*

6.3 Construction in Erasure Model

In this section, we show that Fischlin's scheme $\mathsf{BS_G}$ can be UC secure even in the presence of adaptive adversaries simply by replacing the commitment scheme with a trapdoor commitment scheme. This result is limited to the erasure model where honest users can erase ephemeral randomness when it is no longer needed in computation. It is often said that adaptive security is technically easy with erasures. It is, however, not necessarily obtained for free without impact to the efficiency. Thus it would be worthwhile to formally state that, only with a small modification, Fischlin's generic construction provides universal composability against adaptive adversaries in the erasure model.

Let $\mathsf{BS_T}$ be a stand-alone blind signature scheme $\mathsf{BS_G}$ in Figure 4 with modification such that commitment scheme BC is replaced with a trapdoor commitment scheme TC. The following theorem holds for $\mathsf{BS_T}$.

Proposition 2. *Protocol $\mathsf{Wrap}(\mathsf{BS_T})$ securely realizes $\mathcal{F}_{\mathrm{ncb}}$ in the $\mathcal{F}_{\mathrm{crs}}$-hybrid model with respect to adaptive adversaries in the erasure model.*

Proof. Signature simulatability and unforgeability of $\mathsf{BS_T}$ can be shown in the same way as in the proofs of Lemma 5 and 3, respectively. According to Theorem 4 and Lemma 6, it is thus sufficient to show session equivocability.

Let $\mathsf{SIM.Crs}$ be an algorithm such that, given λ, it computes $(\Sigma_{\mathsf{zk}}, t_{\mathsf{zk}}) \leftarrow \mathsf{NIZK.Crs}(1^\lambda)$, and $(\Sigma_{\mathsf{tc}}, t_{\mathsf{tc}}) \leftarrow \mathsf{TC.Key}(1^\lambda)$ and outputs $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{tc}}), (t_{\mathsf{zk}}, t_{\mathsf{tc}}))$. Clearly this $\mathsf{SIM.Crs}$ is a separable trapdoor generator where $1 = t_{\mathsf{zk}}$ and $2 = t_{\mathsf{tc}}$. Then, as well as Lemma 5, $\mathsf{BS_T}$ is signature simulatable with respect to the generator. Adaptive session simulatability remains to be shown. We construct $\mathsf{SIM.User}$ and $\mathsf{SIM.SesState}$ as follows.

SIM.User: Given $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{tc}}), vk, t_{\mathsf{tc}}; \gamma)$, let $m'||r' \leftarrow \gamma$, compute $(c, z') \leftarrow \mathsf{TC.Com}(\Sigma_{\mathsf{tc}}, m'; r')$, and send c to the signer. On receiving s, compute $\delta \leftarrow \mathsf{SIG.Vrf}(vk, c, s)$, and output δ.

SIM.SesState: Given $((\Sigma_{\mathsf{zk}}, \Sigma_{\mathsf{tc}}), vk, t_{\mathsf{tc}}, \gamma, m, (c||s))$, let $m'||r' \leftarrow \gamma$, compute $r \leftarrow \mathsf{TC.EqOpen}(\Sigma_{\mathsf{tc}}, m, m', r', t_{\mathsf{tc}})$ and output r.

Due to the uniform opening property of TC.EqOpen, the return value r of $\mathcal{O}_0$ distributes identically to that of $\mathcal{O}_1$ and is consistent with c observed through the interaction to the oracles. Thus the above SIM.User and SIM.SesState conforms to session equivocability. ∎

6.4 Construction without Secure Erasures

Protocol $\mathsf{Wrap}(\mathsf{BS}_{\mathsf{T}})$ can be UC secure without erasure if we can construct SIM.SigState for BS_{T}. However, signature equivocability is not generally possible in BS_{T}. Recall that a signature is created by the zero-knowledge simulator in BS_{T}. It therefore can be the case that there exists no randomness that is consistent with a real witness.

To overcome this problem, we consider eliminating the use of a zero-knowledge simulator by providing a correct witness to the proof system through simulation of the signer-side algorithm. Namely, we make the signer's signing algorithm 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 ensures that the same proof could have been created from any other witnesses. Accordingly, a consistent randomness always exists.

To follow the above approach, we use two new building blocks for this construction; a simulatable signature scheme, SSIG (see Appendix B.6), and a state reconstructable non-interactive witness indistinguishable proof system, NIWI (see Appendix B.4). We require two special properties for NIWI. First, it works as a proof of knowledge when the CRS is made in the regular way, which is called the extraction mode. (This property is for proving unforgeability.) Second, when the CRS is made in a special way, called the simulation mode, it allows computation of a randomness that is consistent with a given proof and a witness with extra information observed in creating the proof. (This property is for proving adaptive signature simulatability.) A feasibility result that meets these requirements is shown in [22] under general assumptions.

THE SCHEME BS_{S}. The CRS generation function 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 $\Sigma \leftarrow (\Sigma_{\mathsf{wi}}, \Sigma_{\mathsf{bc}}, \Sigma_{\mathsf{ssig}})$. Key generation function BS.Key is the same as SIG.Key, which outputs vk and sk. The signature generation protocol is shown in Figure 5. In Figure 5, the proof system NIWI proves the following relation between witness $w \leftarrow (s, c, z)$ and instance $x \leftarrow (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m)$:

$$\mathsf{TC.Vrf}(\Sigma_{\mathsf{tc}}, c, m, z) = 1 \wedge \mathsf{SSIG.Vrf}(\Sigma_{\mathsf{ssig}}, vk, c, s) = 1. \tag{15}$$

Verification function 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 $\varphi \leftarrow \mathsf{NIWI.Vrf}(\Sigma_{\mathsf{wi}}, (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m), \sigma)$.

Proposition 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.*

Proof. We first claim that BS_{S} is signature equivocal. Let SIM.Crs be an algorithm such that, given λ, it computes $(\Sigma_{\mathsf{wi}}, t_{\mathsf{wieq}}) \leftarrow \mathsf{NIWI.EqCrs}(1^\lambda)$, $(\Sigma_{\mathsf{tc}}, t_{\mathsf{tc}}) \leftarrow \mathsf{TC.Key}(1^\lambda)$, and $(\Sigma_{\mathsf{ssig}}, t_{\mathsf{ssig}}) \leftarrow$

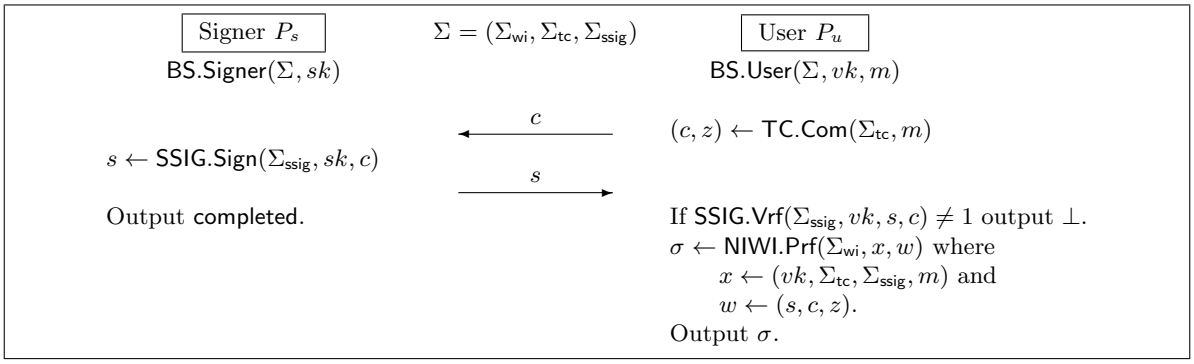

Figure 5: Generic blind signature scheme BS_S. The signature generation protocol.

$\mathsf{SSIG.Crs}(1^\lambda)$ and outputs $\Sigma \leftarrow (\Sigma_{\mathsf{wi}}, \Sigma_{\mathsf{bc}}, \Sigma_{\mathsf{ssig}})$ and a pair of trapdoors $t_1 = (t_{\mathsf{wieq}}, t_{\mathsf{ssig}})$ and $t_2 = (t_{\mathsf{tc}})$. SIM.Crs forms a separable trapdoor generator due to the property of NIWI. We then construct SIM.Sig and SIM.SigState as follows.

SIM.Sig: Given $(\Sigma, vk, m, t_1; \gamma)$, let $r||r'||r'' \leftarrow \gamma$ and compute $(c', z') \leftarrow \mathsf{TC.Com}(\Sigma_{\mathsf{tc}}, m; r)$, $s' \leftarrow \mathsf{SSIG.Sim}(\Sigma_{\mathsf{ssig}}, pk_{\mathsf{ssig}}, c', t_{\mathsf{ssig}}; r')$, and $\sigma \leftarrow \mathsf{NIWI.Prf}(\Sigma_{\mathsf{wi}}, x, w'; r'')$ where the instance is $x = (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m)$ and the witness is $w' = (s', c', z')$. Then output σ.

SIM.SigState: Given $(\Sigma, vk, m, t_1, \xi, r_1, r_2, \theta)$, let $c||s \leftarrow \theta$ and verify that $(c, z) \leftarrow \mathsf{TC.Com}(\Sigma_{\mathsf{tc}}, m; r_2)$ for some z and $\mathsf{SSIG.Vrf}(\Sigma_{\mathsf{ssig}}, vk, c, s) = 1$. If the verification fails, return $\emptyset$. Otherwise, let $r||r'||r'' \leftarrow \xi$ and compute (s', c', z') in the same way as above, and compute $r_1 \leftarrow \mathsf{NIWI.EqState}(\Sigma_{\mathsf{wi}}, t_{\mathsf{wieq}}, x, w', r'', w)$ for $x = (vk, \Sigma_{\mathsf{tc}}, \Sigma_{\mathsf{ssig}}, m)$, $w' = (s', c', z')$, $w = (s, c, z)$. Then output r_1.

According to $\mathcal{O}_0$ in Definition 9, we only need to consider the case where BS.User returns $\sigma \neq \perp$. Recall that $\sigma \neq \perp$ happens only if BS.User receives s that satisfies $1 = \mathsf{SSIG.Vrf}(\Sigma_{\mathsf{ssig}}, vk, s, c)$. Due to the signature simulatability of SSIG, s' computed by SSIG.Sim used in SIM.Sig also satisfies $1 = \mathsf{SSIG.Vrf}(\Sigma_{\mathsf{ssig}}, vk, s', c')$. This means that $w' = (s', c', z')$ forms a proper witness. Furthermore, simulated r_1 distributes uniformly due to the uniform reconstructability of NIWI. Accordingly, the joint distribution of the outputs from SIM.Sig and SIM.SigState in $\mathcal{O}_0$ is identical to that of $\mathcal{O}_1$.

Session equivocability of BS_S holds due to the use of TC as well as the case for BS_T. Then, from Lemma 5, BS_S is equivocal simulation blind. Furthermore, its unforgeability holds just as well as that of BS_G. By applying Theorem 3, we conclude that $\mathsf{Wrap}(\mathsf{BS}_\mathsf{S})$ securely realizes $\mathcal{F}_{\mathrm{ncb}}$. ∎

7 Conclusion

Security notions and related theorems that help in designing and analyzing universally composable non-committing blind signatures were presented. One direction for further research is to find efficient instantiations with stronger security analysis and weaker assumptions. It would also be interesting to see if similar simplification is possible for protocols other than blind signatures without losing their major applications.

References

[1] M. Abe and E. Fujisaki. How to date blind signatures. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96*, volume 1163 of *LNCS*, pages 244–251. Springer-Verlag, 1996.

[2] M. Abe and T. Okamoto. Provably secure partially blind signatures. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer-Verlag, 2000.

[3] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.

[4] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC'03*, LNCS. Springer-Verlag, 2003. (to appear).

[5] S. Brands. Restrictive blinding of secret-key certificates. Technical report, CWI (Centrum voor Wiskunde en Informatica), 1995.

[6] A. B. Buan and K. G. L. Kråkmo. Universally composable blind signatures in the plain model. IACR ePrint Archive 2006/405, 2006.

[7] J. Camenisch, J.-M. Piveteau, and M. Stadler. Blind signatures based on the discrete logarithm problem. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *LNCS*, pages 428–432. Springer-Verlag, 1995.

[8] J. Camenisch, J.-M. Piveteau, and M. Stadler. Fair blind signatures. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *LNCS*, pages 209–219. Springer-Verlag, 1995.

[9] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[10] R. Canetti. On universally composable notions of security for signature, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. `http://eprint.iacr.org`.

[11] R. Canetti. Universally composable signatures, certification and authentication. In *17th Computer Security Foundations Workshop (CSFW)*, 2004. Revised version available in IACR ePrint archive 2003/239.

[12] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Technical Report 2000/067, IACR e-print Archive, 2005. 2nd version updated on 13 Dec 2005.

[13] R. Canetti and M. Fischlin. Universally composable commitments. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer-Verlag, 2001.

[14] R. Canetti and T. Rabin. Universal composition with joint state. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, 2003.

[15] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. Rivest, and A. Sherman, editors, *Advances in Cryptology — Proceedings of Crypto '82*, pages 199–204. Prenum Publishing Corporation, 1982.

[16] D. L. Chaum. Elections with unconditionally-secret ballots and disruptions equivalent to breaking RSA. In C. G. Günther, editor, *Advances in Cryptology — EUROCRYPT '88*, volume 330 of *LNCS*, pages 177–182. Springer-Verlag, 1988.

[17] M. Fischlin. Round-optimal composable blind signatures in the common reference model. In C. Dwork, editor, *Advances in Cryptology — CRYPTO '06*, volume 4117 of *LNCS*, pages 60–77. Springer-Verlag, 2006.

[18] M. Fischlin and D. Schröder. Security of blind signatures under aborts. In *Public Key Cryptography, PKC 2009*, volume 5443 of *LNCS*, pages 297–316. Springer-Verlag, 2009.

[19] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, volume 718 of *LNCS*, pages 244–251. Springer-Verlag, 1993.

[20] J. Garay, A. Kiayias, and H.-S. Zhou. Sound and fine-grain specification of cryptographic tasks. IACR ePrint Archive 2008/132, 2008.

[21] J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer-Verlag, 2006.

[22] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer-Verlag, 2006.

[23] C. Hazay, J. Katz, C. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *Theory of Cryptography Conference, TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer-Verlag, 2007.

[24] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In B. S. Kaliski Jr., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *LNCS*, pages 150–164. Springer-Verlag, 1997.

[25] A. Kiayias and H. Zhou. Concurrent blind signatures without random oracles. In *SCN 2006*, volume 4116 of *LNCS*, pages 49–62. Springer-Verlag, 2006.

[26] A. Kiayias and H. Zhou. Equivocal blind signatures and adaptive uc-security. In R. Canetti, editor, *Theory of Cryptography Conference, TCC 2008*, volume 4948 of *LNCS*, pages 340–355. Springer-Verlag, 2008.

[27] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692. ACM, 2003.

[28] Y. Lindell. Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21(2):200–249, 2008.

[29] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In S. Halevi and T. Rabin, editors, *Theory of Cryptography Conference, TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer-Verlag, 2006. Full version avaialble on ePrint archive.

[30] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):339–360, 2000.

[31] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer-Verlag, 2001.

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

[33] C. P. Schnorr. Security of blind discrete log signatures against interactive attacks. In S. Qing, T. Okamoto, and J. Zhou, editors, *Information and Communications Security*, volume 2229 of *LNCS*, pages 1–12. Springer-Verlag, 2001.

[34] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *LNCS*, pages 533–547. Springer-Verlag, 2002.

Appendices

A Proofs

A.1 Proof of Lemma 1 ($\mathsf{SimBL} \Rightarrow \mathsf{BL}$)

Suppose that BS is not blind, i.e., there exists adversary B^* such that $\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda) = \Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 1) = 1] - \Pr[\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, 0) = 1] > 0$ is not negligible. By using such B^* as a black-box we construct D^* attacking the simulation blindness as follows.

- Given Σ from $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$, flip a coin $b' \leftarrow \{0, 1\}$ and do the following.
  - Execute $(vk, m_0, m_1) \leftarrow B^*(\Sigma)$ and output vk to $\mathbf{SimBL}^{\mathsf{BS}}_{D^*}(\lambda, b)$.
  - Execute $\sigma_{b'} \leftarrow \langle \mathcal{O}_b(m_{b'}), B^* \rangle_L$. Namely, ask $\mathcal{O}_b$ with message $m_{b'}$ and let the succeeding interaction done by B^*.
  - Similarly, execute $\sigma_{1-b'} \leftarrow \langle \mathcal{O}_b(m_{1-b'}), B^* \rangle_L$.
  - If $\sigma_0 = \perp$ or $\sigma_1 = \perp$, then set $\sigma_0 = \sigma_1 = \perp$.
  - Execute $\tilde{b}' \leftarrow B^*(\sigma_1, \sigma_0)$.
- Set $\tilde{b} = 1$ if $\tilde{b}' = b'$, or $\tilde{b} = 0$, otherwise. Output $\tilde{b}$.

Consider the case $b = 1$. Oracle $\mathcal{O}_1$ works faithfully as a user requesting a signature on the given message and return the resulting signature. Thus the view of B^* is in the proper

distribution obtained in $\mathbf{Blind}^{\mathsf{BS}}_{B^*}(\lambda, b')$. Accordingly, under the condition of $b = 1$, we have

$$\begin{aligned}
\Pr[\tilde{b} = 1] &= \Pr[\tilde{b'} = b'] \\
&= \Pr[b' = 1] \cdot \Pr[\tilde{b'} = 1 \mid b' = 1] + \Pr[b' = 0] \cdot \Pr[\tilde{b'} = 0 \mid b' = 0] \\
&= \frac{1}{2} \cdot \Pr[\tilde{b'} = 1 \mid b' = 1] + \frac{1}{2}(1 - \Pr[\tilde{b'} = 1 \mid b' = 0]) \\
&= \frac{1}{2}(\Pr[\tilde{b'} = 1 \mid b' = 1] - \Pr[\tilde{b'} = 1 \mid b' = 0]) + \frac{1}{2} \\
&= \frac{1}{2}\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda) + \frac{1}{2}.
\end{aligned} \tag{16}$$

Next consider the case $b = 0$. In each invocation, oracle $\mathcal{O}_0$ executes $\mathsf{SIM.User}(\Sigma, vk, t)$ that is independent of the input message $m_{b'}$ and $m_{1-b'}$. Furthermore, the simulated signatures are generated by function $\mathsf{SIM.Sig}$ whose input $(\Sigma, vk, \cdot, t)$ is common in both signature generation processes. Thus the resulting signatures are also independent from the view of B^* in the signature generation processes. Accordingly, the entire view of B^* is absolutely independent of b'. Therefore, under the condition of $b = 0$, we have

$$\Pr[\tilde{b} = 1] = \Pr[\tilde{b'} = b'] = \frac{1}{2}. \tag{17}$$

From (16), (17) and the definition of $\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$, we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda) &= \left|\Pr[\tilde{b} = 1 \mid b = 1] - \Pr[\tilde{b} = 1 \mid b = 0]\right| \\
&= \frac{1}{2}\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda) + \frac{1}{2} - \frac{1}{2} \\
&= \frac{1}{2}\mathbf{Adv}^{\mathsf{bl}}_{B^*}(\lambda).
\end{aligned} \tag{18}$$

Accordingly, if BS is simulation blind, i.e. $\mathbf{Adv}^{\mathsf{sib}}_{D^*}(\lambda)$ is negligible for any D^*, no such B^* exists. Thus BS is blind. ∎

Note that, reduction algorithm D^* can provide B^* the information that shows in which session oracle $\mathcal{O}_b$ returns $\perp$. In the case of $b = 1$, it gives B^* a view of selective failure blindness. In the case of $b = 0$, the probability of guessing b' remains $1/2$. Thus we conclude that simulation blindness implies selective failure blindness.

A.2 Proof of Lemma 2 ($\mathsf{BL} \wedge \mathsf{UF} \not\Rightarrow \mathsf{SimBL}$)

Let BS be a blind and unforgeable blind signature scheme in the plain model, e.g., [23]. BS can also be seen as a blind and unforgeable scheme in the CRS model whose CRS is a constant string, say $\emptyset$. Suppose that BS is simulation blind. Then, there exists $\mathsf{SIM.Crs}$ and $\mathsf{SIM.Sig}$ that yield valid signatures for given messages. By using these algorithms, we construct adversary F^* that breaks unforgeability of BS. F^* simply executes $(\emptyset, t) \leftarrow \mathsf{SIM.Crs}(1^\lambda)$ and $\sigma \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t)$ for arbitrary message m and outputs (m, σ). Since $\mathsf{SIM.Crs}$ and $\mathsf{SIM.Sig}$ must be able to simulate the signature generation process between an honest user and honest signer, resulting (m, σ) must be a valid signature with probability close to 1. Thus $\mathbf{Succ}^{\mathsf{uf}}_{F^*}(\lambda) \approx 1$, which is a contradiction. Hence the scheme, BS, is not simulation blind. ∎

A.3 Proof of Lemma 5 ($\mathsf{SesEq} \wedge \mathsf{SigEq} \Rightarrow \mathsf{EqSimBLND}$)

Starting from $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$, we move to $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$ via three transformations and show that each step changes the view of adversary D^* only in negligible manner. Let X_i denote the event that D^* outputs 1 in Game i.

Game 0. This game is the same as $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)$. We thus have

$$\Pr[X_0] = \Pr[\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 1)] \tag{19}$$

by definition.

Game 1. Replace $\mathsf{BS.Crs}(1^\lambda)$ with $\mathsf{SIM.Crs}(1^\lambda)$.

Since $\mathsf{SIM.Crs}$ is a separable trapdoor generator, we have

$$\Pr[X_0] - \Pr[X_1] \leq \mathbf{Adv}^{\mathsf{crs}}_{C^*}(\lambda). \tag{20}$$

Game 2. Modify oracle $\mathcal{O}_1$ in such a way that it takes $t = (t_1, t_2)$ as extra input and does as follows. Let $r_1||r_2 \leftarrow r$. After $\mathsf{BS.User}$ outputs σ, compute $\sigma'[\omega_s] \leftarrow \mathsf{SIM.Sig}(\Sigma, vk, m, t_1)$, and $r'_1 \leftarrow \mathsf{SIM.SigState}(\theta, \omega_s)$ where θ is the transcript observed by $\mathsf{BS.User}$. Then return $(\sigma', r'_1||r_2)$.

Observe that the view of D^* in Game 1 was the same as that of A^* in $\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 1)$. Also observe that modified oracle $\mathcal{O}_1$ in Game 2 works the same as $\mathcal{O}_0$ does in $\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 0)$. Hence the view of D^* in Game 2 is the same as that of A^* in $\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 0)$. We thus have

$$\Pr[X_1] - \Pr[X_2] = \Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 1)] - \Pr[\mathbf{SigEQ}^{\mathsf{BS}}_{A^*}(\lambda, 0)] \leq \mathbf{Adv}^{\mathsf{sigeq}}_{A^*}(\lambda). \tag{21}$$

Game 3. Replace $\mathsf{BS.User}(\Sigma, vk, m)$ in Game 2 with $\mathsf{SIM.User}(\Sigma, vk, t_2)$ that outputs $\delta[\omega_u]$, and set $\sigma = \perp$ if $\delta = 0$. (Note that the transcript θ given to $\mathsf{SIM.SigState}$ can be constructed from ω_u.) Then compute $r'_2 \leftarrow \mathsf{SIM.SesState}(\omega_u, m)$, and return $(\sigma', r'_1||r'_2)$.

We claim that there exists E^* such that

$$\Pr[X_2] - \Pr[X_3] \leq \mathbf{Adv}^{\mathsf{seseq}}_{E^*}(\lambda). \tag{22}$$

The claim can be proven by constructing E^* by using D^* in a straightforward manner. Note that E^* is given trapdoor t_1 and it can compute $\mathsf{SIM.Sig}$ and $\mathsf{SIM.SigState}$ needed to simulate the oracle for D^*.

By wrapping $\mathsf{SIM.SigState}$ and $\mathsf{SIM.SesState}$ into one function $\mathsf{SIM.State}$, Game 3 is now identical to $\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)$. Hence

$$\Pr[X_3] = \Pr[\mathbf{EqSimBL}^{\mathsf{BS}}_{D^*}(\lambda, 0)]. \tag{23}$$

By accumulating (19) to (23), we have

$$\mathbf{Adv}^{\mathsf{eqsib}}_{D^*}(\lambda) \leq \mathbf{Adv}^{\mathsf{crs}}_{C^*}(\lambda) + \mathbf{Adv}^{\mathsf{sigeq}}_{A^*}(\lambda) + \mathbf{Adv}^{\mathsf{seseq}}_{E^*}(\lambda). \tag{24}$$

B Building Blocks

B.1 Commitment Scheme

A commitment scheme BC consists of the following algorithms.

$\Sigma \leftarrow \mathsf{BC.Key}(1^\lambda)$: The commitment key generation algorithm such that outputs a commitment key Σ_{tc} based on the security parameter λ.

$(c, z) \leftarrow \mathsf{BC.Com}(\Sigma, m)$: The commitment function such that, on input message m and the commitment key Σ, outputs a commitment c and an opening information z.

$1/0 \leftarrow \mathsf{BC.Vrf}(\Sigma, c, m, z)$. The verification algorithm such that, on input commitment c and its opening (m, z), outputs 1 for acceptance or 0 for rejection.

Regular stand-alone notion of binding and hiding properties must be fulfilled. The message space is determined by Σ.

B.2 Trapdoor Commitment Scheme

A trapdoor commitment scheme TC consists of the following algorithms.

$(\Sigma, t) \leftarrow \mathsf{TC.Key}(1^\lambda)$: The commitment key generation algorithm such that, on input security parameter λ, generates a commitment key Σ and a trapdoor t.

$(c, z) \leftarrow \mathsf{TC.Com}(\Sigma, m; r)$: The commitment function such that, on input the commitment key Σ and message m, computes a commitment c and the opening information z based on uniform randomness r.

$1/0 \leftarrow \mathsf{TC.Vrf}(\Sigma, c, m, z)$. The verification algorithm such that, on input commitment c and its opening (m, z), outputs 1 for acceptance or 0 for rejection.

$c' \leftarrow \mathsf{TC.Sim}(\Sigma; r')$: The simulation algorithm that outputs a simulated commitment c.

$r \leftarrow \mathsf{TC.EqOpen}(\Sigma, m, m', r', t)$. The opening algorithm that computes a randomness r that is consistent to a commitment made from (m', r') and the target value m. Namely, for $(c, z') \leftarrow \mathsf{TC.Com}(\Sigma, m'; r')$ and $r \leftarrow \mathsf{TC.EqOpen}(\Sigma, m, m', r', t)$, it holds that $(c, z) = \mathsf{TC.Com}(\Sigma, m; r)$ for some z.

While we let $\mathsf{TC.Key}$ to generate the trapdoor for simplicity, it is possible to use a more general formulation that allows two functions for CRS generation; one for genuine CRS and the other for indistinguishable CRS with a trapdoor.

We say that $\mathsf{TC.EqOpen}$ has *uniform opening property* if r distributes uniformly (over proper domain) for any properly generated (Σ, t) and any (m, m') if r distributes uniformly. This property assures that faithfully executed commit and opening phases are perfectly indistinguishable from random commit phase and equivocal opening.

B.3 Non-interactive Zero-Knowledge Proof of Knowledge

A non-interactive zero-knowledge proof of knowledge (for relation R) consists of the following algorithms.

$(\Sigma, t) \leftarrow \mathsf{NIZK.Crs}(1^\lambda)$: A CRS generation algorithm such that, on input the security parameter λ, generates a CRS Σ and a trapdoor t.

$\pi \leftarrow \mathsf{NIZK.Prf}(\Sigma, x, w)$: A prover algorithm such that, on input an instance x and a witness w that satisfies $(x, w) \in R$, outputs a proof π.

$1/0 \leftarrow \mathsf{NIZK.Vrf}(\Sigma, x, \pi)$: A verification algorithm such that, on input an instance x and a proof π, outputs 1 for acceptance or 0 for rejection.

$\pi \leftarrow \mathsf{NIZK.SimPrf}(\Sigma, t, x)$: A zero-knowledge simulator that creates a simulated proof π for input instance x by using trapdoor t.

$w \leftarrow \mathsf{NIZK.Ext}(\Sigma, t, x, \pi)$: A extractor that extracts witness x from the inputs. Relation $(x, w) \in R$ must hold if $1 \leftarrow \mathsf{NIZK.Vrf}(\Sigma, x, \pi)$.

Note that the above formulation is for the same-string zero-knowledge [31], which we used for simplicity. As well as the case of trapdoor commitment, we can use a more general formulation that allows a distinct simulator for $\mathsf{NIZK.Crs}$.

The regular security properties, i.e. completeness, knowledge soundness, and multi-theorem zero-knowledge as defined in [32], must be provided. We also require that the verification algorithm is deterministic for the sake of perfect completeness and consistency of the scheme in Section 5.2.

B.4 State Reconstructable Witness Indistinguishable Proof of Knowledge

A non-interactive witness indistinguishable proof system of knowledge (for relation R) consists of the following algorithms.

$(\Sigma, t) \leftarrow \mathsf{NIWI.Crs}(1^\lambda)$: The CRS generation algorithm for the extraction mode such that, on input the security parameter λ, generates a CRS Σ and a trapdoor t.

$\pi \leftarrow \mathsf{NIWI.Prf}(\Sigma, x, w; r)$: The prover algorithm such that, on input an instance x and an witness that satisfies $(x, w) \in R$, outputs a proof π.

$1/0 \leftarrow \mathsf{NIWI.Vrf}(\Sigma, x, \pi)$: The verification algorithm such that, on input an instance x and a proof π, outputs 1 for acceptance or 0 for rejection.

$w \leftarrow \mathsf{NIWI.Ext}(\Sigma, t, x, \pi)$: The extractor that extracts witness x from the inputs. $(x, w) \in R$ must hold if $1 \leftarrow \mathsf{NIWI.Vrf}(\Sigma, x, \pi)$.

$(\Sigma', t') \leftarrow \mathsf{NIWI.EqCrs}(1^\lambda)$: The CRS generation algorithm for the simulation mode such that, on input the security parameter λ, generates a CRS Σ' and a trapdoor t'.

$r \leftarrow \mathsf{NIWI.EqState}(\Sigma', t', x, w', r', w)$: The state reconstruction algorithm in the simulation mode that computes randomness r such that $\mathsf{NIWI.Prf}(\Sigma, x, w; r) = \mathsf{NIWI.Prf}(\Sigma, x, w'; r')$.

We require NIWI to provide the following properties.

- (CRS Indistinguishability.) Let $(\Sigma, t) \leftarrow \mathsf{NIWI.Crs}(1^\lambda)$ and $(\Sigma', t') \leftarrow \mathsf{NIWI.EqCrs}(1^\lambda)$. Then Σ and Σ' are indistinguishable.
- (Uniform Reconstractability.) If r' distributes uniformly over $\mathcal{R}$, so does r computed by $r \leftarrow \mathsf{NIWI.EqState}(\Sigma', t', x, w', r', w)$.

Here, $\mathcal{R}$ is the randomness space for function $\mathsf{NIWI.Prf}$ defined by Σ'.

B.5 Digital Signature Scheme

A signature scheme SIG consists of three algorithms.

$(pk, sk) \leftarrow \mathsf{SIG.Key}(1^\lambda)$: The key generation algorithm that generates a verification key pk and a signing key sk based on the security parameter λ.

$s \leftarrow \mathsf{SIG.Sign}(sk, m)$: The signature generation algorithm that computes a signature s for input message m by using signing key sk.

$1/0 \leftarrow \mathsf{SIG.Vrf}(pk, m, s)$. The verification algorithm that outputs 1 for accept or 0 for rejection according to the input.

Standard completeness condition and existential unforgeability must hold.

B.6 Simulatable Signature Scheme

A simulatable signature scheme SSIG in the CRS model consists of following algorithms.

$(\Sigma, t) \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$: The CRS generation algorithm such that, on input security parameter λ, outputs a common reference string Σ and a trapdoor t.

$(pk, sk) \leftarrow \mathsf{SSIG.Key}(\Sigma)$: The key generation algorithm that generates a verification key pk and a signing key sk.

$s \leftarrow \mathsf{SSIG.Sign}(\Sigma, sk, m)$: The signature generation algorithm that computes a signature s for input message m by using signing key sk.

$s \leftarrow \mathsf{SSIG.Sim}(\Sigma, pk, m, t)$: Another signature generation algorithm that computes a signature s for message m by using trapdoor t.

$1/0 \leftarrow \mathsf{SSIG.Vrf}(\Sigma, pk, m, s)$. The verification algorithm that outputs 1 for accept or 0 for rejection according to the input.

Completeness and existential unforgeability against chosen message attacks are defined in the same way as for regular signature schemes with trivial modifications to adapt to the CRS model. Roughly speaking, signature simulatability is that if a signer that generates a (potentially bogus) public-key can create valid signatures then $\mathsf{SSIG.Sim}$ should be able to output valid signatures on the same messages. More formally, for any $(\Sigma, t) \leftarrow \mathsf{SSIG.Crs}(1^\lambda)$, and for any $(pk, m_1, s_1, \ldots, m_k, s_k) \leftarrow G^*(\Sigma)$, if $1 \leftarrow \mathsf{SSIG.Vrf}(\Sigma, pk, m_i, s_i)$ then $s'_i \leftarrow \mathsf{SSIG.Sim}(\Sigma, pk, m_i, t)$ fulfills $1 \leftarrow \mathsf{SSIG.Vrf}(\Sigma, pk, m_i, s'_i)$ for all $1 \geq i \geq k$. 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.

B.7 A Note on Instantiation

All the building blocks in Appendix B can be instantiated in a bilinear setting. Let $\Lambda := (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$ be a description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ of prime order q equipped with efficient bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It also includes a random generator g of $\mathbb{G}_1$ and $\tilde{g}$ of $\mathbb{G}_2$. We use Λ as a common group for our building blocks and relative CRS and key generation functions are modified to take Λ as input instead of the security parameter and generate keys and other elements based on Λ.

For proof systems defined in Appendix B.3 and B.4, the Groth-Sahai proof system [22] (the GS proof system for short) meets the requirements under SXDH or DLIN assumption. The proof system 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.

For a trapdoor commitment scheme in Appendix B.2 (and a standard commitment scheme in Appendix B.1), one may use the following construction. $\mathsf{TC.Key}$ takes group description Λ as input. It selects $\tau \in \mathbb{Z}_q$, sets $h \leftarrow g^\tau$, and outputs commitment key $\Sigma_{\mathsf{tc}} \leftarrow (\Lambda, h)$ and trapdoor τ. $\mathsf{TC.Com}$ takes commitment key Σ_{tc} and a message $m \in \mathbb{Z}_q$ as input. It computes $c = g^m h^r \in \mathbb{G}_1$ and $z = \tilde{g}^r \in \mathbb{G}_2$ for random $r \xleftarrow{\$} \mathbb{Z}_q$ and outputs (c, z). $\mathsf{TC.Vrf}$ takes (c, m, z) as input and output 1 (for accept) if $e(c/g^m, \tilde{g}) = e(h, z)$. It outputs 0 (for rejection), otherwise. Finally, $\mathsf{TC.EqOpen}$ takes (m, m', r', τ) and outputs $r \leftarrow r' + (m' - m)/\tau$. To see the correctness observe that $e(c/g^m, \tilde{g}) = e(h^r, \tilde{g}) = e(h, \tilde{g}^r) = e(h, z)$. holds for correct commitment $c = g^m h^r$ and its opening information $z = \tilde{g}^r$. Also for commitment $c = g^{m'} h^{r'}$ and simulated open information $z \leftarrow \tilde{g}^r = \tilde{g}^{r' + (m' - m)/\tau}$, it holds that $e(h, z) = e(h, \tilde{g}^{r' + (m' - m)/\tau}) = e(g^{m' - m} h^{r'}, \tilde{g}) = e(c/g^m, \tilde{g})$. Hence z is a correct opening information for c and m. The scheme is perfectly hiding and computationally binding under (S)XDH assumption. The distribution of Σ_{tc}, c, z output from $\mathsf{TC.Key}$, $\mathsf{TC.Sim}$, $\mathsf{TC.EqOpen}$ are statistically close to uniform.

Finally, the simulatable signature scheme in Appendix B.6 (and the signature scheme in Appendix B.5) must be able to sign group elements and the verification predicate must be represented as a product of pairings to fit to other building blocks described above. For such a signature scheme a feasibility result based on DLIN can be seen in [21].