Universally Composable Blind Signatures in the Plain Model

Aslak Bakke Buan, Kristian Gjøsteen, Lillian Kråkmo

November 9, 2006

Abstract

In the universal composability framework, we define an ideal functionality for blind signatures, as an alternative to a functionality recently proposed by Fischlin. Fischlin proves that his functionality cannot be realized in the plain model, but this result does not apply to our functionality. We show that our functionality is realized in the plain model by a blind signature protocol if and only if the corresponding blind signature scheme is secure with respect to blindness and non-forgeability, as defined by Juels, Luby and Ostrovsky.

Keywords: Blind signatures, universal composability

1 Introduction

The idea of blind signatures was proposed by Chaum [4] as a key ingredient for anonymous electronic cash applications. Blind signatures allow a bank to issue signatures without seeing the content of the signed documents, and at the same time prevent users from forging signatures. Pointcheval and Stern [7] first defined *non-forgeability* for blind signature schemes. Juels, Luby and Ostrovsky (JLO) [6] further formalized the concept by giving a general definition of a blind signature scheme and formulating the required security properties: *blindness* and non-forgeability. Informally, a scheme has blindness if it is infeasible for a malicious signer to determine the order of which two messages are signed by interaction with an honest user. A scheme has non-forgeability if, given l interactions with an honest signer, it is infeasible for a malicious user to produce more than l valid signatures.

Universally composable (UC) security is a framework proposed by Canetti [2] as a way to define security for protocols such that security-preserving composition is possible. This allows for a modular design and analysis of protocols. For each cryptographic task, an *ideal functionality* can be defined, which incorporates the required properties of a protocol for the task and the allowed actions of an adversary. A protocol is said to *securely realize* the ideal functionality if, loosely speaking, any effect caused by an adversary attacking the protocol can be obtained by an adversary attacking the ideal functionality. When designing complex protocols, one can allow the involved parties to have secure access to ideal functionalities. Then, when implementing the protocol, each ideal functionality is replaced by a protocol securely realizing the functionality. The *composition theorem* then guarantees security. We refer to [2] for a complete overview of this framework.

Since UC security is a powerful and useful notion, an interesting question is how it relates to conventional security notions. A recent paper by Fischlin [5] addresses this question in the context of blind signatures. The author defines an ideal functionality for blind signatures, $\mathcal{F}_{\mathrm{BlSig}}$, and shows that blind signature schemes realizing $\mathcal{F}_{\mathrm{BlSig}}$ in the plain model do not exist. He does this by showing that $\mathcal{F}_{\mathrm{BlSig}}$ can be used to realize the functionality $\mathcal{F}_{\mathrm{com}}$ for commitment schemes, then applying a well-known impossibility result [3]. To realize $\mathcal{F}_{\mathrm{BlSig}}$, Fischlin has to work in the common reference string model.

One somewhat awkward artefact of Fischlin's functionality is that any realizing functionality must encode the entire message to be signed into the first protocol message. This restricts any realizing protocols to a maximal message length, otherwise blindness would be violated. We could extend this by signing a hash of the message instead of the message itself. If we use a collision resistant hash function, this clearly does not degrade the real security of any such scheme. Unfortunately, the modified protocol no longer realizes the functionality. This can be considered an artefact of the universal composability framework, not of the specific functionality, but nonetheless, it is undesirable and we would like to allow blind signature protocols that do not encode the entire message to be signed into the first protocol message.

To achieve this, we propose a new blind signature functionality. The main change is that while Fischlin's functionality requires, even for corrupt users, that the message to be signed is a part of the signing command given to the ideal functionality, our functionality does not look at the message specified by a corrupt user, but instead gives him a "free signature". Only when message/signature pairs are verified, the functionality can learn what message was signed. As a consequence, it can not be used to realize $\mathcal{F}_{\mathrm{com}}$, meaning that Fischlin's impossibility result does not apply to our functionality. Even so, our functionality still captures the essentials of blind signature schemes.

We can now prove that a blind signature protocol realizes $\mathcal{F}_{\mathrm{BS}}$ in the plain model if and only if the corresponding blind signature scheme satisfies weak blindness and non-forgeability, as originally defined by JLO. We note that, in this paper, we refer to JLO's version of blindness as *weak blindness*, reflecting the fact that the adversary is not allowed to choose his target keys.

On the negative side, our functionality requires the signer to be honest during key generation. In the UC framework, this corresponds to the property of weak blindness, that the adversary does not choose his own key. We believe that in some cases this correctly models the real world, e.g. in a scenario where the key generation for a bank is performed by a financial supervisory authority.

As an alternative to this corruption model, we may allow corruption of the signer before the key generation takes place, but demand that the signer reveals his public and secret keys to a trusted third party for verification. A possible scenario is one where the bank generates its own keys, but then shows them to a financial supervisory authority. We express this in the UC framework by adding an uncorruptible trusted third party to the protocol, to whom the signer sends his public and secret keys (over a confidential channel) before any user issues signing requests. With this requirement, it is clear that our result also holds for a slightly stronger version of blindness, where the adversary chooses his target keys, but then reveals them to the simulator.

If we are willing to return to the common reference string model, it should be possible to relax this requirement even further, by including in the public key a commitment to the secret key along with a proof that this commitment is correct. (In the CRS model with a carefully chosen commitment scheme, the simulator can extract the secret key, essentially reducing everything to the case of blindness.) In this case, the main advantage of our functionality compared to Fischlin's functionality is that we allow protocols where the user's first message does not contain an extractable copy of the message to be signed. This means that the functionality could be realized by blind signature protocols accepting messages of arbitrary length.

Another, minor difference between the functionalities is that our functionality lets the environment decide whether or not the signer should grant a signature to a user. This is a vital property if the functionality is to be used in a bigger protocol, but it could of course be added to Fischlin's functionality.

Our main contribution in this paper is a more flexible blind signature functionality that allows a larger class of realizing protocols, while still capturing the essence of blind signatures.

In Sect. 2 of this paper, we review the properties of a blind signature scheme and give formal definitions of blindness and non-forgeability. In Sect. 3 we present our ideal functionality for blind signatures and prove our main result.

2 Blind Signatures

Our definition of a blind signature scheme corresponds to the one given by Juels, Luby and Ostrovsky in [6].

Definition 1 (Blind Signature Scheme)**.** *A blind signature scheme $\mathcal{BS}$ is a four-tuple (Signer, User, Gen, Verify) with the following properties:*

- *Gen is a probabilistic polynomial time algorithm, which takes as input a security parameter τ (encoded as 1^τ) and outputs a pair (pk, sk) of public and secret keys.*
- *Signer and User are a pair of polynomially-bounded probabilistic interactive Turing machines, given as common input a public key pk. In addition, Signer is given a corresponding secret key sk, and User is given a message m. The length of all inputs must be polynomial in the security parameter τ. Signer and User interact according to the protocol. At the end of the interaction, Signer outputs either completed or not completed and User outputs either fail or $\sigma(m)$.*
- *Verify is a deterministic polynomial time algorithm, which takes as input a public key pk, a message m and a signature $\sigma(m)$, and outputs either accept or reject, indicating whether $\sigma(m)$ is a valid signature on the message m.*

It is required that for any message m, and for all key pairs (pk, sk) output by Gen, if both Signer and User follow the protocol, then Signer(pk, sk) outputs completed, User(pk, m) outputs $\sigma(m)$, and Verify$(pk, m, \sigma(m))$ outputs accept.

The *security* of blind signature schemes is formally defined below. We note that, throughout this paper, *weak blindness* corresponds to the original definition of blindness given by JLO.

Definition 2 (Weak Blindness)**.** *Consider the experiment $\mathbf{Exp}^{wb}_{\mathcal{BS},A}(\tau)$ (steps 1, 2, ..., 6) in Fig. 1, where A is an algorithm. We define the advantage of A in breaking $\mathcal{BS}$ with respect to* weak blindness *as*

$$\boldsymbol{Adv}^{wb}_{\mathcal{BS},A}(\tau) = \Big| Pr\left[b' = 1 | b = 1\right] - Pr\left[b' = 1 | b = 0\right] \Big|.$$

The scheme $\mathcal{BS}$ is said to be secure with respect to weak blindness if, for all probabilistic polynomial time A, $\boldsymbol{Adv}^{wb}_{\mathcal{BS},A}(\tau)$ is negligible in τ.

$\mathbf{Exp}^{\mathrm{wb/b}}_{\mathcal{BS},A}(\tau)$:

1. $(pk, sk) \leftarrow Gen(1^\tau)$. Run A on input $(1^\tau, pk, sk)$.

1'. Run A on input 1^τ. $(pk, sk) \leftarrow A$.

2. $(m_0, m_1) \leftarrow A$.
3. $b \leftarrow \{0, 1\}$.
4. Let A engage in two parallel interactive protocols, the first with $User(pk, m_b)$ and the second with $User(pk, m_{1-b})$.
5. If the first $User$ outputs $\sigma(m_b)$ and the second $User$ outputs $\sigma(m_{1-b})$, then give $\{\sigma(m_0), \sigma(m_1)\}$ to A as additional input.
6. A outputs a bit b'.

$\mathbf{Exp}^{\mathrm{wror/ror}}_{\mathcal{BS},A}(\tau)$:

1. $(pk, sk) \leftarrow Gen(1^\tau)$. Run A on input $(1^\tau, pk, sk)$.

1' Run A on input 1^τ. $(pk, sk) \leftarrow A$.

2. $b \leftarrow \{0, 1\}$.
3. A polynomial (in τ) number of times, A is allowed to output a message m_1:

   If $b = 0$, choose a random message m_0 and let A engage in a protocol with $User(pk, m_0)$. Run a protocol between $Signer(pk, sk)$ and $User(pk, m_1)$ to get $\sigma(m_1)$. If $User(pk, m_0)$ outputs $\sigma(m_0)$, give $\sigma(m_1)$ to A as additional input.

   If $b = 1$, let A engage in a protocol with $User(pk, m_1)$. If $User(pk, m_1)$ outputs $\sigma(m_1)$, give $\sigma(m_1)$ to A as additional input.
4. A outputs a bit b'.

$\mathbf{Exp}^{\mathrm{nf}}_{\mathcal{BS},A}(\tau)$:

1. $(pk, sk) \leftarrow Gen(1^\tau)$.
2. Let $A(1^\tau, pk)$ engage in polynomially many (in τ) parallel interactive protocols, with polynomially many (in τ) copies of $Signer(pk, sk)$, where A decides in an adaptive manner when to stop. Let l be the number of executions, where the $Signer$ outputs *completed*.
3. A outputs a collection $\{(m_1, \sigma(m_1)), \ldots, (m_k, \sigma(m_k))\}$, subject to the constraint that $(m_i, \sigma(m_i)) \neq (m_j, \sigma(m_j))$ for $1 \leq i < j \leq k$, and $Verify(pk, m_i, \sigma(m_i))$ outputs *accept* for $1 \leq i \leq k$.

Figure 1: Experiments for defining blindness and non-forgeability.

We now introduce a slightly stronger definition, *blindness*, in which the adversary determines the key pair (pk, sk), and hands it over to us.

Definition 3 (Blindness)**.** *Consider the experiment* $\mathbf{Exp}^{b}_{\mathcal{BS},A}(\tau)$ *(steps* $1'$*,* 2*, ...,* 6*) in Fig. 1, where A is an algorithm. We define the advantage of A in breaking* $\mathcal{BS}$ *with respect to* blindness *as*

$$\boldsymbol{Adv}^{b}_{\mathcal{BS},A}(\tau) = \Big| Pr\left[b' = 1 | b = 1\right] - Pr\left[b' = 1 | b = 0\right] \Big| .$$

The scheme $\mathcal{BS}$ *is said to be secure with respect to blindness if, for all probabilistic polynomial time A,* $\boldsymbol{Adv}^{b}_{\mathcal{BS},A}(\tau)$ *is negligible in* τ*.*

An even stronger notion, *strong blindness*, is defined in the same manner, except that the adversary is only required to output a public key pk in the first step of the experiment.

Definition 4 (Non-forgeability)**.** *Consider the experiment* $\mathbf{Exp}^{nf}_{\mathcal{BS},A}(\tau)$ *in Fig. 1, where A is an algorithm. We define the success rate of A in breaking* $\mathcal{BS}$ *with respect to* non-forgeability *as*

$$\boldsymbol{Succ}^{nf}_{\mathcal{BS},A}(\tau) = Pr\left[k > l\right] .$$

The scheme $\mathcal{BS}$ *is said to be secure with respect to non-forgeability if, for all probabilistic polynomial time A,* $\boldsymbol{Succ}^{nf}_{\mathcal{BS},A}(\tau)$ *is negligible in* τ*.*

We now present another notion for blindness, adapting the notion "real-or-random" for symmetric encryption given in [1]. The idea is that, when interacting with an honest user, it should be infeasible for a malicious signer to tell whether a known message or a hidden random string is being signed. (Note that there does not seem to be a natural "strong" version of this notion.)

Definition 5 ((Weak) Real-or-Random Blindness)**.** *Consider the experiment* $\mathbf{Exp}^{wror/ror}_{\mathcal{BS},A}(\tau)$ *in Fig. 1, where A is an algorithm. We define the advantage of A in breaking* $\mathcal{BS}$ *with respect to* (weak) real-or-random blindness *as*

$$\boldsymbol{Adv}^{wror/ror}_{\mathcal{BS},A}(\tau) = \Big| Pr\left[b' = 1 | b = 1\right] - Pr\left[b' = 1 | b = 0\right] \Big| .$$

The scheme $\mathcal{BS}$ *is said to be secure with respect to real-or-random blindness if, for all probabilistic polynomial time A,* $\boldsymbol{Adv}^{wror/ror}_{\mathcal{BS},A}(\tau)$ *is negligible in* τ*.*

Adapting of a result from [1], we obtain the following theorem, the proof of which is given in Appendix A:

Theorem 1. *A blind signature scheme* $\mathcal{BS}$ *is secure with respect to (weak) blindness if and only if it is secure with respect to (weak) real-or-random blindness.*

3 Universally Composable Blind Signatures

Our ideal functionality for blind signatures, $\mathcal{F}_{\mathrm{BS}}$, is defined in Figure 2. The protocol $\pi_{\mathcal{BS}}$ is defined in Figure 3.

For ease of presentation, the session id (SID), which should be present in all messages, is not included. The first message sent to the functionality must be (**KeyGen**). We require that no corruption takes place before the (**KeyGen**) message has been processed, which amounts to the key generation being honest. In addition to generating keys, the ideal adversary $\mathcal{S}$ produces signature generation and verification facilities for the functionality: $\Pi(m)$ simulates a conversation between honest signer and honest user, producing a signature σ on m. $\pi(m, \sigma)$ outputs 1 if σ is a valid signature on m, and 0 otherwise.

When the signer receives a signature request from a user, the environment determines whether or not the user is entitled to a signature. If the functionality is to be used in a bigger protocol, allowing this decision to depend on outer circumstances may be useful. For instance, it may depend on the balance of the user's bank account.

In the plain model, parties running a protocol have authenticated communication channels, but messages are potentially delayed by the adversay. To handle this, we let $\mathcal{S}$ delay messages between parties. $\mathcal{S}$ also decides when to inform the respective parties about the outcome of a signature request. This allows $\mathcal{S}$ to get the order of these messages right, that is, according to the real protocol.

Our functionality keeps track of signatures generated by corrupt users by means of a *free signature count*. In Fischlin's functionality, when a corrupted user produces a signature by interaction with an honest signer, this signature is stored together with the message m input by the user. However, since the user is corrupt, we do not know if m really is the message being signed. Assume that the user instead signs m', a message never seen by $\mathcal{F}_{\text{BlSig}}$, and obtains a valid pair (m', σ). Upon verification, (m', σ) will be rejected by $\mathcal{F}_{\text{BlSig}}$, while accepted by the real protocol. Our functionality overcomes this problem, by increasing the free signature count every time an honest signer completes a protocol with a corrupt user. If the free signature count is more than zero, a pair (m, σ) may be accepted upon verification, even if $\mathcal{F}_{\text{BS}}$ has never seen m before. We argue that $\mathcal{F}_{\text{BS}}$ still incorporates the required properties of a blind signature protocol, as it still prevents a user from obtaining more valid signatures than generated by interaction with an honest signer.

The protocol $\pi_{\mathcal{BS}}$ is described in Figure 3. The user initiates a protocol, and upon receiving the first message from a user, the signer lets the environment decide whether he should engage in the protocol, analogously to the formulation of $\mathcal{F}_{\text{BS}}$. We now prove our main result:

Theorem 2. *The blind signature scheme $\mathcal{BS}$ is secure with respect to weak blindness and non-forgeability if and only if the protocol $\pi_{\mathcal{BS}}$ securely realizes $\mathcal{F}_{BS}$ in the plain model.*

Proof. Only if: Assuming that the blind signature scheme $\mathcal{BS}$ is secure with respect to weak blindness and non-forgeability, we show that for every adversary A interacting with parties running $\pi_{\mathcal{BS}}$ in the plain model, there is an ideal adversary $\mathcal{S}$ such that no environment Z can tell whether it is interacting with A and $\pi_{\mathcal{BS}}$ or with $\mathcal{S}$ and the ideal protocol $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$.

As usual, $\mathcal{S}$ runs a simulated copy of A, and forwards all messages from Z to A and back. When A corrupts a party P, $\mathcal{S}$ corrupts $\tilde{P}$. When $\tilde{P}$ is corrupt, any input from Z meant for $\tilde{P}$ goes directly to $\mathcal{S}$, who forwards it to A on the input tape corresponding to P, and the other way around. Moreover, $\mathcal{S}$ can send messages to $\mathcal{F}_{\text{BS}}$ in the name of $\tilde{P}$, and messages from $\mathcal{F}_{\text{BS}}$ meant for $\tilde{P}$ go to $\mathcal{S}$. $\mathcal{S}$ runs as described below.

In the following, when a party P controls another party P', the notation "P/P'" should be read as "P, in the name of P'".

Algorithm $\mathcal{S}(\tau)$:

- Upon receiving (**KeyGen**, Q) from $\mathcal{F}_{\text{BS}}$, $\mathcal{S}$ runs $Gen(\tau)$, obtains a key pair (pk, sk) and stores

$\mathcal{F}_{\text{BS}}$ proceeds as follows, with signer $\tilde{Q}$, users $\tilde{P}_1, \ldots, \tilde{P}_n$ and an ideal adversary $\mathcal{S}$.

On message (**KeyGen**) from the signer $\tilde{Q}$:

1. Send (**KeyGen**) to $\mathcal{S}$ and wait.
2. Upon receipt of $(\textbf{Key}, pk, \Pi, \pi)$ from $\mathcal{S}$, store (pk, Π, π), send $(\textbf{Key}, pk)$ to $\tilde{Q}$ and stop.

On message $(\textbf{Sign}, pk, m)$ from a user $\tilde{P}_i$:

1. Send $(\textbf{Sign}, P_i)$ to $\mathcal{S}$.
2. Upon receipt of $(\textbf{Sign}, P_i, \mathit{ack})$ from $\mathcal{S}$, send $(\textbf{Sign}, P_i?)$ to $\tilde{Q}$.
3. Upon receipt of $(\textbf{Sign}, P_i, \mathit{denied})$ from $\tilde{Q}$, send $(\textbf{Sign}, P_i, \mathit{denied})$ to $\mathcal{S}$. Wait for $(\textbf{Sign}, P_i, \mathit{denied}, \mathit{ack})$ from $\mathcal{S}$, and output $(\textbf{Sign}, \mathit{denied})$ to $\tilde{P}_i$.
4. Upon receipt of $(\textbf{Sign}, P_i)$ from $\tilde{Q}$, send $(\textbf{Sign}, P_i)$ to $\mathcal{S}$.
    1. Upon receipt of $(\textbf{Signature}, P_i, Q\ \mathit{completed})$ from $\mathcal{S}$, send $(\textbf{Signature}, P_i)$ to $\tilde{Q}$. If $\tilde{Q}$ is honest and $\tilde{P}_i$ is corrupt, increase the free signature count.
    2. Upon receipt of $(\textbf{Signature}, P_i, Q\ \mathit{not\ completed})$ from $\mathcal{S}$, send $(\textbf{Signature}, P_i, \mathit{not\ completed})$ to $\tilde{Q}$.
    3. Upon receipt of $(\textbf{Signature}, P_i, P_i\ \mathit{completed})$ from $\mathcal{S}$, if P_i is honest, $\sigma \overset{r}{\leftarrow} \Pi(m)$. If $(m, \sigma, 0)$ is stored, then stop. Otherwise, store $(m, \sigma, 1)$ and send $(\textbf{Signature}, \sigma)$ to $\tilde{P}_i$.
    4. Upon receipt of $(\textbf{Signature}, P_i, P_i\ \mathit{fail})$ from $\mathcal{S}$, send $(\textbf{Signature}, \mathit{fail})$ to $\tilde{P}_i$.

On message $(\textbf{Verify}, pk, m, \sigma)$ from an honest user $\tilde{P}_i$:

1. If $(m, \sigma, 1)$ is stored, send (**Verify**) to $\tilde{P}_i$ and stop.
2. If $(m, \sigma, 0)$ is stored, send $(\textbf{Verify}, \mathit{reject})$ to $\tilde{P}_i$ and stop.
3. If $\pi(m, \sigma) = 1$ and Q is corrupt, store $(m, \sigma, 1)$, send (**Verify**) to $\tilde{P}_i$ and stop.
4. If $\pi(m, \sigma) = 1$ and the free signature count is larger than zero, decrease the signature count, store $(m, \sigma, 1)$, send (**Verify**) to $\tilde{P}_i$ and stop.
5. Store $(m, \sigma, 0)$ and send $(\textbf{Verify}, \mathit{reject})$ to $\tilde{P}_i$ and stop.

Figure 2: Blind signature functionality $\mathcal{F}_{\text{BS}}$

$\pi_{\mathcal{BS}}$ is defined as follows, with signer $Q(\tau)$ and users $P_1, \ldots, P_n$.

$Q(\tau)$:

1. Upon the first input (**KeyGen**), run the algorithm $Gen(\tau)$, obtain a key pair (pk, sk) and output $(\textbf{Key}, pk)$.
2. Upon receiving (x) from P_i, send $(\textbf{Sign}, P_i?)$ to Z and wait.
3. Upon input $(\textbf{Sign}, P_i, \textit{denied})$ from Z, send $(\textbf{Sign}, \textit{denied})$ to P_i and stop.
4. Upon input $(\textbf{Sign}, P_i)$ from Z, run *Signer* and give it x as input on its communication tape. Forward any output on *Signer*'s communication tape to P_i, and vice versa. If $Signer(pk, sk)$ outputs *completed*, then output $(\textbf{Signature}, P_i)$. Otherwise, if *Signer* outputs *not completed*, output $(\textbf{Signature}, P_i, \textit{not completed})$.

P_i:

1. Upon input $(\textbf{Sign}, pk, m)$, run $User(pk, m)$. When $User(pk, m)$ outputs x on its communication tape, send (x) to Q and wait
2. Upon receiving $(\textbf{Sign}, \textit{denied})$ from Q, output $(\textbf{Sign}, \textit{denied})$.
3. Upon receiving (y) from Q, forward y to *User*'s communication tape. Forward any output on *User*'s communication tape to Q, and vice versa. If *User* outputs σ, then output $(\textbf{Signature}, \sigma)$. Otherwise, if *User* outputs *fail*, output $(\textbf{Signature}, \textit{fail})$.
4. Upon input $(\textbf{Verify}, pk, m, \sigma)$, run the algorithm *Verify*, obtain *accept*/*reject* and output $(\textbf{Verify})/(\textbf{Verify}, \textit{reject})$.

Figure 3: Blind signature protocol $\pi_{\mathcal{BS}}$

(pk, sk). $\mathcal{S}$ then produces signature generation and verification facilities Π and π, and sends $(\textbf{Key}, Q, pk, \Pi, \pi)$ to $\mathcal{F}_{\text{BS}}$.

- Upon receiving $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\text{BS}}$, when both Q and P_i are honest: $\mathcal{S}$ simulates honest Q and honest P_i, and allows A to delay any communication between $\mathcal{S}/Q$ and $\mathcal{S}/P_i$.

    1. $\mathcal{S}$ chooses a random message $\tilde{m}$, and runs $User(pk, \tilde{m})$. When $User(pk, \tilde{m})$ outputs x on its communication tape, $\mathcal{S}/P_i$ sends (x) to $\mathcal{S}/Q$. When $\mathcal{S}/Q$, receives (x), $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \textit{ack})$ to $\mathcal{F}_{\text{BS}}$.
    2. Upon receipt of $(\textbf{Sign}, P_i, \textit{denied})$ from $\mathcal{F}_{\text{BS}}$, $\mathcal{S}/Q$ sends $(\textbf{Sign}, \textit{denied})$ to $\mathcal{S}/P_i$. When $\mathcal{S}/P_i$ receives $(\textbf{Sign}, \textit{denied})$, $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \textit{denied}, \textit{ack})$ to $\mathcal{F}_{\text{BS}}$.
    3. Upon receipt of $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\text{BS}}$, $\mathcal{S}$ runs $Signer(pk, sk)$ and gives it x as input on its communication tape. $\mathcal{S}/Q$ forwards any output on *Signer*'s communication tape to $\mathcal{S}/P_i$, and vice versa. Similarily, $\mathcal{S}/P_i$ forwards any output on *User*'s communication tape to $\mathcal{S}/Q$, and vice versa.
    4. If *Signer* outputs *completed*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, Q\ \textit{completed})$ to $\mathcal{F}_{\text{BS}}$. Otherwise, if *Signer* outputs *not completed*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, Q\ \textit{not completed})$ to $\mathcal{F}_{\text{BS}}$.
    5. If *User* outputs σ, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, P_i\ \textit{completed})$ to $\mathcal{F}_{\text{BS}}$. Otherwise, if *User* outputs *fail*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, P_i\ \textit{fail})$ to $\mathcal{F}_{\text{BS}}$.

- Upon receiving $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\mathrm{BS}}$, when Q is corrupt and P_i is honest: $\mathcal{S}$ simulates honest P_i.

  1. $\mathcal{S}$ chooses a random message $\tilde{m}$, and runs $User(pk, \tilde{m})$. When $User(pk, \tilde{m})$ outputs x on its communication tape, $\mathcal{S}/P_i$ sends (x) to A.
  2. $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \mathit{ack})$ to $\mathcal{F}_{\mathrm{BS}}$, and $\mathcal{S}/\tilde{Q}$ receives $(\textbf{Sign}, P_i?)$ from $\mathcal{F}_{\mathrm{BS}}$.
  3. If $\mathcal{S}/P_i$ receives $(\textbf{Sign}, \mathit{denied})$ from A, then $\mathcal{S}/\tilde{Q}$ sends $(\textbf{Sign}, P_i, \mathit{denied})$ to $\mathcal{F}_{\mathrm{BS}}$. Upon receiving $(\textbf{Sign}, P_i, \mathit{denied})$ from $\mathcal{F}_{\mathrm{BS}}$, $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \mathit{denied}, \mathit{ack})$ to $\mathcal{F}_{\mathrm{BS}}$.
  4. If $\mathcal{S}/P_i$ receives (y) from A, then $\mathcal{S}/\tilde{Q}$ sends $(\textbf{Sign}, P_i)$ to $\mathcal{F}_{\mathrm{BS}}$, and $\mathcal{S}$ receives $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\mathrm{BS}}$. $\mathcal{S}/P_i$ forwards y to *User*'s communication tape. $\mathcal{S}/P_i$ forwards any output on *User*'s communication tape to A, and vice versa.
  5. If *User* outputs σ, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, P_i\ \mathit{completed})$ to $\mathcal{F}_{\mathrm{BS}}$. Otherwise, if *User* outputs *fail*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, P_i\ \mathit{fail})$ to $\mathcal{F}_{\mathrm{BS}}$.
  6. $\mathcal{S}$ sends $(\textbf{Signature}, P_i, Q\ \mathit{not\ completed})$ to $\mathcal{F}_{\mathrm{BS}}$, and $\mathcal{S}/\tilde{Q}$ receives $(\textbf{Signature}, P_i, Q\ \mathit{not\ completed})$ from $\mathcal{F}_{\mathrm{BS}}$.

- When Q is honest and P_i is corrupt: $\mathcal{S}$ simulates honest Q.

  1. When $\mathcal{S}/Q$ receives (x) from A, $\mathcal{S}$ chooses a random message $\tilde{m}$ and $\mathcal{S}/\tilde{P}_i$ sends $(\textbf{Sign}, pk, \tilde{m})$ to $\mathcal{F}_{\mathrm{BS}}$. Upon receipt of $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\mathrm{BS}}$, $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \mathit{ack})$ to $\mathcal{F}_{\mathrm{BS}}$.
  2. Upon receipt of $(\textbf{Sign}, P_i, \mathit{denied})$ from $\mathcal{F}_{\mathrm{BS}}$, $\mathcal{S}/Q$ sends $(\textbf{Sign}, \mathit{denied})$ to A. $\mathcal{S}$ sends $(\textbf{Sign}, P_i, \mathit{denied}, \mathit{ack})$ to $\mathcal{F}_{\mathrm{BS}}$, and $\mathcal{S}/\tilde{P}_i$ receives $(\textbf{Sign}, \mathit{denied})$ from $\mathcal{F}_{\mathrm{BS}}$.
  3. Upon receipt of $(\textbf{Sign}, P_i)$ from $\mathcal{F}_{\mathrm{BS}}$, $\mathcal{S}$ runs $Signer(pk, sk)$ and gives it x as input on its communication tape. $\mathcal{S}/Q$ forwards any output on *Signer*'s communication tape to A, and vice versa.
  4. If *Signer* outputs *completed*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, Q\ \mathit{completed})$ to $\mathcal{F}_{\mathrm{BS}}$. Otherwise, if *Signer* outputs *not completed*, then $\mathcal{S}$ sends $(\textbf{Signature}, P_i, Q\ \mathit{not\ completed})$ to $\mathcal{F}_{\mathrm{BS}}$.
  5. $\mathcal{S}$ sends $(\textbf{Signature}, P_i, P_i\ \mathit{fail})$ to $\mathcal{F}_{\mathrm{BS}}$, and $\mathcal{S}/\tilde{P}$ receives $(\textbf{Signature}, \mathit{fail})$ from $\mathcal{F}_{\mathrm{BS}}$.

We want to show that if there exists an environment Z able to distinguish $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{BS}}}$ and $\pi_{\mathcal{BS}}$ with non-negligible probability, then we can construct an adversary A^{ror} breaking the real-or-random blindness or an adversary A^{nf} breaking the non-forgeability of our scheme. To this end, we will consider a series of games, where we gradually modify the behaviour of the involved parties.

Game 0**:** In this game, Z interacts with the protocol $\pi_{\mathcal{BS}}$ running with parties $Q, P_1, \ldots, P_n$.

Game 1**:** This game is the same as Game 0, with the following modifications: The honest parties are now simulated by P. In addition, P keeps track of the produced signatures: If an honest P_i generates a signature σ on a message m, P stores $(m, \sigma, 1)$. However, if P_i is corrupt and Q is honest, and Q outputs $(\textbf{Signature}, P_i)$, P increases the free signature count.

From Z's point of view, there is no difference between Game 0 and Game 1. Therefore, if we let G_i denote the output of Z when taking part in Game i, we have

$$\Big|\Pr[G_0 = 1] - \Pr[G_1 = 1]\Big| = 0.$$

Game 2: This game is the same as Game 1, with the following modifications: If an honest P_i generates a signature σ on a message m, and $(m, \sigma, 0)$ is stored, then P stops. Upon input $(\textbf{Verify}, pk, m, \sigma)$ to an honest P_i, P responds as follows:

1. If $(m, \sigma, 1)$ is stored, send $(\textbf{Verify})$ to P_i and stop.
2. If $(m, \sigma, 0)$ is stored, send $(\textbf{Verify}, \textit{reject})$ to P_i and stop.
3. If $\pi(m, \sigma) = 1$ and Q is corrupt, store $(m, \sigma, 1)$, send $(\textbf{Verify})$ to P_i and stop.
4. If $\pi(m, \sigma) = 1$ and the free signature count is larger than zero, decrease the signature count, store $(m, \sigma, 1)$, send $(\textbf{Verify})$ to P_i and stop.
5. Store $(m, \sigma, 0)$ and send $(\textbf{Verify}, \textit{reject})$ to P_i and stop.

We now prove the following lemma:

Lemma 1. *There is an adversary A^{nf} such that*

$$\Big| Pr[G_1 = 1] - Pr[G_2 = 1] \Big| \leq \boldsymbol{Succ}^{nf}_{\mathcal{BS}, A^{nf}}(\tau).$$

Proof. We simplify the proof by defining the event F in a game as follows:

F: At some point during the game, while Q is honest, some honest party P_i is asked to verify a valid pair (m, σ), but there is no recorded entry $(m, \sigma, 1)$, and the free signature count is zero.

We note that, unless F occurs, Game 1 and Game 2 proceed identically. In particular, if it ever happens in Game 2 that an honest P_i generates a signature σ on a message m, and $(m, \sigma, 0)$ is stored, then $(m, \sigma, 0)$ must have been stored while Q was honest, which implies that F occurred. Hence we have

$$\Big| \Pr[G_1 = 1] - \Pr[G_2 = 1] \Big| \leq \Pr[F].$$

We now construct an adversary A^{nf} trying to break the non-forgeability of our scheme. $A^{\text{nf}}(1^\tau, pk)$ runs simulated copies of Z, A and the corrupted parties. A^{nf} also simulates P, and behaves exactly as P, with the following exceptions: When Q is asked to generate keys, instead of running *Gen*, Q simply outputs $(\textbf{Key}, pk)$. When Q engages in a protocol with some P_i, A^{nf} engages in a protocol with $\textit{Signer}(pk, sk)$, forwarding inputs from P_i to $\textit{Signer}(pk, sk)$ and vice versa. If F occurs, we deduce that for some numbers k, l such that $k > l$, Z has produced k valid pairs (m, σ), while Q has output *completed* l times. Since A^{nf} stores the valid signatures and controls the free signature count, A^{nf} can detect F and output the k valid pairs (m, σ). Hence we have

$$\textbf{Succ}^{\text{nf}}_{\mathcal{BS}, A^{\text{nf}}}(\tau) \geq \Pr[F],$$

which leads to

$$\Big| \Pr[G_1 = 1] - \Pr[G_2 = 1] \Big| \leq \textbf{Succ}^{\text{nf}}_{\mathcal{BS}, A^{\text{nf}}}(\tau),$$

by which the proof is complete. □

Game 3: In this game, Z interacts with the ideal protocol $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$ running with parties $\tilde{P}_1, \ldots, \tilde{P}_n$.

We observe that Game 2 and Game 3 differs only in the case where P_i is honest: When signing a message m, in Game 2, Q runs a protocol with $\mathit{User}(pk, m)$, while in Game 3, Q runs a protocol with $\mathit{User}(pk, \tilde{m})$, where $\tilde{m}$ is a randomly chosen message.

We prove the following lemma:

Lemma 2. *There is an adversary A^{ror} such that*

$$\left| Pr[G_2 = 1] - Pr[G_3 = 1] \right| = \boldsymbol{Adv}^{ror}_{\mathcal{BS}, A^{ror}}(\tau).$$

Proof. We construct an adversary A^{ror} trying to break the real-or-random blindness of our scheme: $A^{\text{ror}}(1^\tau, pk, sk)$ runs simulated copies of Z, A and the corrupted parties. A^{ror} also simulates P. We note that, since we require that Q is honest during the key generation, A^{ror} controls Q at this stage. When Q is asked to generate keys, A^{ror} simply outputs $(\textbf{Key}, pk)$ in the name of Q. A^{ror} behaves exactly as P, with the following exception: When Q engages in a protocol to sign some message m_1 input by some honest user P_i, A^{ror} outputs m_1 and engages in a protocol with $\mathit{User}(pk, m_b)$, where b is A^{ror}'s challenge bit and m_0 is a randomly chosen message. A^{ror} forwards inputs from Q to $\mathit{User}(pk, m_b)$ and the other way around. If A^{ror} obtains σ as additional input, then A^{ror} outputs $(\textbf{Signature}, \sigma)$ in the name of P_i. Finally, when Z outputs a bit b', A^{ror} outputs b'.

We observe that, if A^{ror}'s challenge bit $b = 0$, then Z's environment in Game 3 is perfectly simulated, while if $b = 1$, Z's environment in Game 2 is perfectly simulated. Hence, we compute A^{ror}'s advantage as follows:

$$\begin{aligned}\textbf{Adv}^{\text{ror}}_{\mathcal{BS}, A^{\text{ror}}}(\tau) &= \left| \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] \right| \\ &= \left| \Pr[G_2 = 1] - \Pr[G_3 = 1] \right|,\end{aligned}$$

and the proof is complete. □

By a standard hybrid argument, if there exists an effective distinguisher Z between Game 0 and Game 3, then, for some i, $0 \leq i \leq 2$, there exists an effective distinguisher Z_i between Game i and Game $i+1$. As shown above, such a Z_i would imply the existence of an effective A^{nf} or an effective A^{ror}. Hence this part of the proof is complete, and we proceed with the opposite direction.

If: We start by assuming that there exists an effective adversary A^{ror}, and construct an environment Z trying to distinguish interaction with $\pi_{\mathcal{BS}}$ from interaction with $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$. Z runs as follows:

1. Z sends $(\textbf{KeyGen})$ to Q, and obtains pk.
2. Z corrupts Q and obtains sk.
3. Z runs $A^{\text{ror}}(\tau, pk, sk)$.
4. Each time A^{ror} outputs a message m_1, Z sends $(\textbf{Sign}, pk, m_1)$ to P_1. When Z/Q receives $(\textbf{Sign})$ from P_1, Z/Q sends $(\textbf{Sign})$ to P_1. Z now lets A^{ror} run a protocol with P_1 on Q's behalf. If P_1 outputs $(\textbf{Signature}, \sigma)$, Z gives σ to A^{ror} as additional input.
5. When A^{ror} outputs a bit b', Z outputs b'.

We note that, if Z interacts with $\pi_{\mathcal{BS}}$, A^{ror}'s environment in $\mathbf{Exp}^{\text{ror}}_{\mathcal{BS},A^{\text{ror}}}(\tau)$ in the case where $b = 1$ is perfectly simulated, while if Z interacts with $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$, this holds for the case where $b = 0$. So we get

$$\begin{aligned}\Big|\Pr[Z_{\pi_{\mathcal{BS}}} = 1] - \Pr[Z_{\text{IDEAL}_{\mathcal{F}_{\text{BS}}}} = 1]\Big| &= \Big|\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\Big| \\ &= \mathbf{Adv}^{\text{ror}}_{\mathcal{BS},A^{\text{ror}}}(\tau),\end{aligned}$$

which completes this part of the proof.

Finally, assuming that there exists an effective adversary A^{nf}, we construct an environment Z trying to distinguish interaction with $\pi_{\mathcal{BS}}$ from interaction with $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$. Z runs as follows:

1. Z sends (**KeyGen**) to Q, and obtains pk.
2. Z runs $A^{\text{nf}}(\tau, pk)$.
3. Each time A^{nf} engages in a protocol, Z corrupts a party P_i and lets A^{nf} run a protocol with Q on P_i's behalf. (When Q outputs $(\mathbf{Sign}, P_i?)$, Z replies with $(\mathbf{Sign}, P_i)$).
4. Assume that Q outputs l messages on the form $(\mathbf{Signature}, P_i)$ for some i, and that A^{nf} outputs k valid pairs (m_i, σ_i). If $k > l$, Z sends $(\mathbf{Verify}, pk, m_i, \sigma_i)$ to an honest party P_j for each $i = 1, \ldots, k$. For $i = k$, if P_j replies with $(\mathbf{Verify}, \mathit{reject})$, then Z outputs 0. Otherwise, if P_2 replies with (**Verify**), then Z outputs 1.

We note that, if Z interacts with $\pi_{\mathcal{BS}}$, then A^{nf}'s environment in $\mathbf{Exp}^{\text{nf}}_{\mathcal{BS},A}(\tau)$ is perfectly simulated, and if A^{nf} outputs $k > l$ valid pairs then Z's output will be 1. If Z interacts with $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$, then A^{nf} does not run in its expected environment, but whatever happens, we know that Z's output will be 0 in this case.

If A^{nf} outputs $k > l$ valid pairs (m_i, σ_i), then if Z interacts with $\pi_{\mathcal{BS}}$, Z's output will be 1, while if Z interacts with $\text{IDEAL}_{\mathcal{F}_{\text{BS}}}$, Z's output will be 0. So we get

$$\begin{aligned}\Big|\Pr[Z_{\pi_{\mathcal{BS}}} = 1] - \Pr[Z_{\text{IDEAL}_{\mathcal{F}_{\text{BS}}}} = 1]\Big| &= \Big|\Pr[k > l] - 0\Big| \\ &= \mathbf{Succ}^{\text{nf}}_{\mathcal{BS},A^{\text{nf}}}(\tau),\end{aligned}$$

by which the proof is complete. □

References

[1] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 394–403, Washington, DC, USA, 1997. IEEE Computer Society.

[2] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Cryptology ePrint Archive, Report 2000/067, 2005. Available at `http://eprint.iacr.org/2000/067`.

[3] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 19–40, London, UK, 2001. Springer-Verlag.

[4] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology-Crypto'82*, pages 199–203, 1982.

[5] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *Advances in Cryptology-Crypto 2006*. Springer-Verlag, 2006.

[6] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 150–164, London, UK, 1997. Springer-Verlag.

[7] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In *ASIACRYPT '96: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 252–265, London, UK, 1996. Springer-Verlag.

A Proof of Theorem 1

In order to simplify the proof, we introduce the notion left-or-right blindness, which is similar to weak blindness, except that the step where A outputs two messages (m_0, m_1) and interacts with $User(pk, m_b)$ and $User(pk, m_{1-b})$ can be repeated polynomially many times.

Definition 6 (Left-or-Right Blindness)**.** *Consider the following experiment, where A is a probabilistic polynomial-time algorithm which controls Signer but not User:*
$\mathbf{Exp}^{lor}_{\mathcal{BS},A}(\tau)$*:*

1. $(pk, sk) \leftarrow Gen(\tau)$.
2. $b \leftarrow \{0,1\}$.
3. *Run* $A(1^\tau, pk, sk)$.
4. *When A outputs two messages (m_0, m_1): Let A engage in two parallel interactive protocols, the first with $User(pk, m_b)$ and the second with $User(pk, m_{1-b})$. If the first User outputs $\sigma(m_b)$ and the second User outputs $\sigma(m_{1-b})$, then give $\{\sigma(m_b), \sigma(m_{1-b})\}$ to A as additional input, ordered according to the corresponding order of (m_0, m_1). This step can be repeated polynomially many (in τ) times.*
5. *A outputs a bit b'.*

We define the advantage of A in breaking $\mathcal{BS}$ with respect to left-or-right blindness as

$$\boldsymbol{Adv}^{lor}_{\mathcal{BS},A}(\tau) = \Big| Pr\big[b' = 1 | b = 1\big] - Pr\big[b' = 1 | b = 0\big]\Big|.$$

The scheme $\mathcal{BS}$ is said to be secure with respect to left-or-right blindness if, for all A, $\boldsymbol{Adv}^{lor}_{\mathcal{BS},A}(\tau)$ is negligible in τ.

The proof consists of showing the following relations among the security notions for blind signature schemes:

1. Weak Blindness $\Rightarrow$ left-or-right blindness
2. Left-or-right blindness $\Rightarrow$ real-or-random blindness
3. Real-or-random blindness $\Rightarrow$ left-or-right-blindness
4. Left-or-right blindness $\Rightarrow$ weak blindness

Weak Blindness $\Rightarrow$ Left-or-Right Blindness: Assume that there exists an effective adversary A^{lor} with respect to left-or-right blindness. We will construct an effective adversary A^{wb} with respect to weak blindness, using A^{lor} as a subroutine. In more detail, we consider the experiment $\mathbf{Exp}^{\text{wb}}_{\mathcal{BS},A^{\text{wb}}}(\tau)$, where A^{wb} proceeds as given below.

Algorithm $A^{\text{wb}}(1^\tau, pk, sk)$:

1. $j \leftarrow \{0,\dots,n\}$, where n is the number of times that A^{lor} repeats step 4 in $\mathbf{Exp}^{\text{lor}}_{\mathcal{BS},A^{\text{lor}}}(\tau)$.
2. Run A^{lor} with input $(1^\tau, pk, sk)$.
3. Denote by $(m_{00}, m_{10}),\dots,(m_{0(n-1)}, m_{1(n-1)})$ the message pairs output by A^{lor} during the run. When A^{lor} outputs (m_{0i}, m_{1i}):
   - If $0 \le i < j$, let A^{lor} engage in two parallel interactive protocols, the first with $User(pk, m_{0i})$ and the second with $User(pk, m_{1i})$. If $User(pk, m_{0i})$ outputs $\sigma(m_{0i})$ and $User(pk, m_{1i})$ outputs $\sigma(m_{1i})$, then A^{lor} is given $(\sigma(m_{0i}), \sigma(m_{1i}))$ as additional input.
   - If $i = j$, then output (m_{0i}, m_{1i}). Forward inputs from $User(pk, m_{bi})$ and $User(pk, m_{(1-b)i})$ to the respective communication tapes of A^{lor}, and the other way around. If $(\sigma(m_{0i}), \sigma(m_{1i}))$ is obtained as additional input, forward this to A^{lor}.
   - If $j < i < n$, let A^{lor} engage in two parallel interactive protocols, the first with $User(pk, m_{1i})$ and the second with $User(pk, m_{0i})$. If $User(pk, m_{0i})$ outputs $\sigma(m_{0i})$ and $User(pk, m_{1i})$ outputs $\sigma(m_{1i})$, then A^{lor} is given $(\sigma(m_{0i}), \sigma(m_{1i}))$ as additional input.
4. When A^{lor} outputs a bit b', output b'.

For $0 \le i \le n$, define P_i as the probability that A^{lor} outputs 1 given that, for the first i message pairs (m_0, m_1) output during a run, A^{wb} lets A^{lor} interact with $User(pk, m_0)$ and $User(pk, m_1)$, respectively. Then, for the last $n - i$ message pairs, A^{wb} lets A^{lor} interact with $User(pk, m_1)$ and $User(pk, m_0)$, respectively.

We note that if $i = 0$, A^{wb} simulates A^{lor}'s environment in $\mathbf{Exp}^{\text{lor}}_{\mathcal{SC},A^{\text{lor}}}(\tau)$ in the case where $b = 1$, and if $i = n$, A^{wb} simulates A^{lor}'s environment in the case where $b = 0$. So by assumption we have

$$\Big| P_0 - P_n \Big| = \delta(\tau)$$

for some $\delta(\tau)$ non-negligible in τ. We can write

$$\Big| (P_0 - P_1) + (P_1 - P_2) + \cdots + (P_{n-1} - P_n) \Big| = \delta(\tau).$$

We note that if A^{wb}'s challenge bit $b=0$, the probability that A^{lor} outputs 1 is P_{j+1}, and if $b=1$, the probability that A^{ror} outputs 1 is P_j. Hence we get

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{wb}}_{\mathcal{BS},A^{\mathrm{wb}}}(\tau) &= \Big|\Pr\big[b'=1|b=1\big]-\Pr\big[b'=1|b=0\big]\Big| \\
&= \Big|\Pr\big[b'=1|b=1\wedge j=0\big]\cdot\Pr\left[j=0\right]+\cdots+ \\
&\quad \Pr\big[b'=1|b=1\wedge j=n-1\big]\cdot\Pr\left[j=n-1\right] \\
&\quad -\Pr\big[b'=1|b=0\wedge j=0\big]\cdot\Pr\left[j=0\right]-\cdots- \\
&\quad \Pr\big[b'=1|b=0\wedge j=n-1\big]\cdot\Pr\left[j=n-1\right]\Big| \\
&= \Big|\frac{1}{n}\left(P_0+\cdots+P_{n-1}-(P_1+\cdots+P_n)\right)\Big| \\
&= \frac{1}{n}\Big|(P_0-P_1)+\cdots+(P_{n-1}-P_n)\Big| \\
&= \frac{\delta(\tau)}{n}.
\end{aligned}$$

Since $\delta(\tau)$ is non-negligible in τ, and n is polynomial in τ, $\frac{1}{n}\delta(\tau)$ is also non-negligible in τ.

Left-or-Right Blindness $\Rightarrow$ Real-or-Random Blindness: In this part, assuming that there exists an effective adversary A^{ror} with respect to real-or-random blindness, we consider $\mathbf{Exp}^{\mathrm{lor}}_{\mathcal{BS},A^{\mathrm{lor}}}(\tau)$, where A^{lor} runs the following procedure.

Algorithm $A^{\mathrm{lor}}(1^\tau, pk, sk)$:

1. Run A^{ror} with input $(1^\tau, pk, sk)$.
2. Each time A^{ror} outputs a message m_1:
   - Choose a random message m_0 and output (m_0, m_1).
   - Forward inputs from $User(pk, m_b)$ to A^{ror}'s communication tape, and the other way around. Run the protocol with $User(pk, m_{1-b})$ honestly (to ensure that a signature is produced). If $(\sigma(m_0), \sigma(m_1))$ is obtained as additional input, forward $\sigma(m_1)$ to A^{ror}.
3. When A^{ror} outputs a bit b', output b'.

We observe that when A^{lor}'s challenge bit $b=0$, A^{lor} simulates A^{ror}'s environment in $\mathbf{Exp}^{\mathrm{ror}}_{\mathcal{BS},A^{\mathrm{ror}}}(\tau)$ in the case where A^{ror}'s challenge bit $b=0$, and correspondingly for the case $b=1$. Hence we get

$$\mathbf{Adv}^{\mathrm{lor}}_{\mathcal{BS},A^{\mathrm{lor}}}(\tau) = \mathbf{Adv}^{\mathrm{ror}}_{\mathcal{BS},A^{\mathrm{ror}}}(\tau),$$

which, by assumption, is non-negligible in τ.

Real-or-Random blindness $\Rightarrow$ Left-or-Right blindness: Assuming that there exists an effective adversary A^{lor} with respect to left-or-right blindness, our goal is to construct an adversary A^{ror} breaking the real-or-random blindness of our scheme. To simplify this part of the proof, we define the following games:

Game 1: This game is the same as $\mathbf{Exp}^{\mathrm{lor}}_{\mathcal{BS},A}(\tau)$, except that we always have $b=0$.

Game 2: This game is the same as Game 1, with the following modifications: Each time A outputs a message pair, say (m_0, m'_1), we choose a random message m_1, and let A engage in protocols with $\mathit{User}(pk, m_0)$ and $\mathit{User}(pk, m_1)$, respectively. Moreover, we run internally a protocol between honest $\mathit{Signer}(pk, sk)$ and honest $\mathit{User}(pk, m'_1)$, and obtain $\sigma(m'_1)$. If $\mathit{User}(pk, m_0)$ outputs $\sigma(m_0)$ and $\mathit{User}(pk, m_1)$ outputs $\sigma(m_1)$, then we give $(\sigma(m_0), \sigma(m'_1))$ to A as additional input.

Game 3: This game is the same as Game 1, with the following modifications: Each time A outputs a message pair, say (m'_0, m'_1), we choose two random messages m_0 and m_1, and let A engage in protocols with $\mathit{User}(pk, m_0)$ and $\mathit{User}(pk, m_1)$, respectively. Moreover, we run two protocols internally, one between honest $\mathit{Signer}(pk, sk)$ and honest $\mathit{User}(pk, m'_0)$, the other between honest $\mathit{Signer}(pk, sk)$ and honest $\mathit{User}(pk, m'_1)$. We thus obtain $\sigma(m'_0)$ and $\sigma(m'_1)$. If $\mathit{User}(pk, m_0)$ outputs $\sigma(m_0)$ and $\mathit{User}(pk, m_1)$ outputs $\sigma(m_1)$, then we give $(\sigma(m'_0), \sigma(m'_1))$ to A as additional input.

We start by constructing a distinguisher between Game 1 and Game 3, A^{13}, using A^{lor} as a subroutine. That is, A^{13} participates in either Game 1 or Game 3 and outputs a bit b.

Algorithm $A^{13}(1^\tau, pk, sk)$:

1. Run A^{lor} with input $(1^\tau, pk, sk)$.
2. $\tilde{b} \leftarrow \{0, 1\}$
3. Each time A^{lor} outputs a message pair, say $(\tilde{m}_0, \tilde{m}_1)$:
   - Output $(\tilde{m}_{\tilde{b}}, \tilde{m}_{1-\tilde{b}})$.
   - Engage in two protocols, say with User_0 and User_1, respectively. Forward outputs from User_0 and User_1 to the respective communication tapes of A^{lor}, and the other way around. If some pair $(\sigma(\tilde{m}_0), \sigma(\tilde{m}_1))$ is obtained as additional input, forward this to A^{lor}.
4. When A^{lor} outputs a bit b', if $b' = \tilde{b}$, then output 1, otherwise output 0.

We observe that, if A^{13} participates in Game 1, A^{13} simulates A^{lor}'s environment in $\mathbf{Exp}^{\mathrm{lor}}_{\mathcal{BS},A^{\mathrm{lor}}}(\tau)$, with $\tilde{b}$ acting as the challenge bit. On the other hand, if A^{13} participates in Game 3, A^{lor} gets no information about $\tilde{b}$, so in this case A^{13} outputs 1 with probability $\frac{1}{2}$. In line with the former definitions, we compute the advantage of A^{13} as follows, where b'' denotes the output of A^{13}:

$$\mathbf{Adv}^{13}_{\mathcal{BS},A}(\tau) = \Big|\Pr[b'' = 1|\text{Game 1}] - \Pr[b'' = 1|\text{Game 3}]\Big|,$$

where

$$\begin{aligned}\Pr[b'' = 1|\text{Game 1}] &= \Pr[b' = 1|\text{Game 1} \wedge \tilde{b} = 1] \cdot \Pr[\tilde{b} = 1] \\ &\quad + \Pr[b' = 0|\text{Game 1} \wedge \tilde{b} = 0] \cdot \Pr[\tilde{b} = 0] \\ &= \Pr[b' = 1|\text{Game 1} \wedge \tilde{b} = 1] \cdot \frac{1}{2} \\ &\quad + \Big(1 - \Pr[b' = 1|\text{Game 1} \wedge \tilde{b} = 0]\Big) \cdot \frac{1}{2}.\end{aligned}$$

So we get

$$\begin{aligned}\mathbf{Adv}^{13}_{\mathcal{BS},A}(\tau) &= \Big|\frac{1}{2}\Big(\Pr[b' = 1|\text{Game } 1 \wedge \tilde{b} = 1] - \Pr[b' = 1|\text{Game } 1 \wedge \tilde{b} = 0]\Big) + \frac{1}{2} - \frac{1}{2}\Big| \\ &= \frac{1}{2} \cdot \mathbf{Adv}^{\text{lor}}_{\mathcal{BS},A}(\tau),\end{aligned}$$

which, by assumption, is non-neglible in τ. This means that A^{13} is an effective distinguisher between Game 1 and Game 3.

By a standard hybrid argument, if there exists an effective A^{13}, then there exists an effective distinguisher A^{12} between Game 1 and Game 2 or an effective distinguisher A^{23} between Game 2 and Game 3. We complete this part of the proof by showing that, using either A^{12} or A^{23}, we can construct an effective adversary A^{ror} with respect to real-or-random blindness.

First, we assume that there exists an effective A^{12} and consider the experiment $\mathbf{Exp}^{\text{ror}}_{\mathcal{BS},A^{\text{ror}}}(\tau)$, where A^{ror} proceeds as described below.

Algorithm $A^{\text{ror}}(1^\tau, pk, sk)$:

1. Run A^{12} with input $(1^\tau, pk, sk)$.
2. When A^{12} outputs a message pair, say $(\tilde{m}_0, \tilde{m}_1)$, output $\tilde{m}_1$.
3. Simulate honest $User(pk, \tilde{m}_0)$ internally, and engage in a protocol, say with $User_1$. Forward inputs from $User(pk, \tilde{m}_0)$ and $User_1$ to the respective communication tapes of A^{12}, and the other way around. If $User(pk, \tilde{m}_0)$ outputs $\sigma(\tilde{m}_0)$ and $\sigma(\tilde{m}_1)$ is obtained as additional input, forward $(\sigma(\tilde{m}_0), \sigma(\tilde{m}_1))$ to A^{12}.
4. When A^{12} outputs a bit b', output b'.

We note that if A^{ror}'s challenge bit $b = 0$, then A^{ror} simlulates A^{12}'s environment in Game 2, while if $b = 1$, A^{ror} simlulates A^{12}'s environment in Game 1. We compute the advantage of A^{ror} as follows:

$$\begin{aligned}\mathbf{Adv}^{\text{ror}}_{\mathcal{BS},A}(\tau) &= \Big|\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\Big| \\ &= \Big|\Pr[b' = 1|\text{Game } 1] - \Pr[b' = 1|\text{Game } 2]\Big| \\ &= \mathbf{Adv}^{12}_{\mathcal{BS},A}(\tau),\end{aligned}$$

which, by assumption, is non-negligible in τ.

Now, in a similar manner, we assume that there exists an effective A^{23} and consider the experiment $\mathbf{Exp}^{\text{ror}}_{\mathcal{BS},A^{\text{ror}}}(\tau)$, where A^{ror} proceeds as follows:

Algorithm $A^{\text{ror}}(1^\tau, pk, sk)$:

1. Run A^{23} with input $(1^\tau, pk, sk)$.
2. When A^{23} outputs a message pair, say $(\tilde{m}_0, \tilde{m}_1)$, output $\tilde{m}_0$.
3. Choose randomly a message $\tilde{m}$. Simulate internally $User(pk, \tilde{m})$ in an honest manner, and engage in a protocol, say with $User_0$. Forward inputs from $User_1$ and $User(pk, \tilde{m})$ to the respective communication tapes of A^{23}, and the other way around. Moreover, simulate internally a protocol between honest $Signer(pk, sk)$ and honest $User(pk, \tilde{m}_0)$, so that a signature

$\sigma(\tilde{m}_0)$ is obtained. If $User(pk, \tilde{m})$ outputs $\sigma(\tilde{m})$ and $\sigma(\tilde{m}_0)$ is obtained as additional input, forward $(\sigma(\tilde{m}_0), \sigma(\tilde{m}_1))$ to A^{23}.

4. When A^{23} outputs a bit b', output b'.

We note that if A^{ror}'s challenge bit $b = 0$, then A^{ror} simlulates A^{23}'s environment in Game 3, while if $b = 1$, A^{ror} simlulates A^{23}'s environment in Game 2. We compute the advantage of A^{ror} as follows:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ror}}_{\mathcal{BS},A}(\tau) &= \Big|\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]\Big| \\ &= \Big|\Pr[b' = 1|\text{Game } 2] - \Pr[b' = 1|\text{Game } 3]\Big| \\ &= \mathbf{Adv}^{23}_{\mathcal{BS},A}(\tau),\end{aligned}$$

which, by assumption, is non-negligible in τ.

Left-or-Right blindness $\Rightarrow$ weak blindness: Assume that there exists an effective adversary A^{wb} with respect to weak blindness, and consider the experiment $\mathbf{Exp}^{\mathrm{lor}}_{\mathcal{BS},A^{\mathrm{lor}}}(\tau)$, where A^{lor} proceeds as described below.

Algorithm $A^{\mathrm{lor}}(1^\tau, pk, sk)$:

1. Run A^{wb} with input $(1^\tau, pk, sk)$.
2. When A^{wb} outputs a message pair (m_0, m_1), output (m_0, m_1).
3. Forward inputs from $User(pk, m_b)$ and $User(pk, m_{1-b})$ to the respective communication tapes of A^{wb}, and the other way around. If $(\sigma(m_0), \sigma(m_1))$ is obtained as additional input, forward this to A^{wb}.
4. When A^{wb} outputs a bit b', output b'.

It is clear that when A^{lor}'s challenge bit $b = 0$, A^{lor} simulates A^{wb}'s environment in $\mathbf{Exp}^{\mathrm{wb}}_{\mathcal{BS},A^{\mathrm{wb}}}(\tau)$ in the case where A^{wb}'s challenge bit $b = 0$, and correspondingly for the case $b = 1$. Hence we get

$$\mathbf{Adv}^{\mathrm{lor}}_{\mathcal{BS},A^{\mathrm{lor}}}(\tau) = \mathbf{Adv}^{\mathrm{wb}}_{\mathcal{BS},A^{\mathrm{wb}}}(\tau),$$

which, by assumption, is non-negligible in τ. $\square$