A preliminary version appears in Public-Key Cryptography — PKC 2009, Lecture Notes in Computer Science, Volume 5443, pp. 297-316, Springer-Verlag, 2009.

Security of Blind Signatures Under Aborts and Applications to Adaptive Oblivious Transfer

Marc Fischlin Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de
marc.fischlin@gmail.com schroeder@me.com

Abstract. We explore the security of blind signatures under aborts where the user or the signer may stop the interactive signature issue protocol prematurely. Several works on blind signatures discuss security only in regard of completed executions and usually do not impose strong security requirements in case of aborts. One of the exceptions is the paper of Camenisch, Neven and shelat (Eurocrypt 2007) where the notion of selective-failure blindness has been introduced. Roughly speaking, selective-failure blindness says that blindness should also hold in case the signer is able to learn that some executions have aborted.

Here we augment the work of Camenisch et al. by showing how to turn every secure blind signature scheme into a selective-failure blind signature scheme. Our transformation only requires an additional computation of a commitment and therefore adds only a negligible overhead. We also study the case of multiple executions and notions of selective-failure blindness in this setting. We then discuss the case of user aborts and unforgeability under such aborts. We show that every three-move blind signature scheme remains unforgeable under such user aborts. Together with our transformation for selective-failure blindness we thus obtain an easy solution to ensure security under aborts of either party and which is applicable for example to the schemes of Pointcheval and Stern (Journal of Cryptology, 2000).

We finally revisit the construction of Camenisch et al. for simulatable adaptive oblivious transfer protocols, starting from selective-failure blind signatures where each message only has one valid signature (uniqueness). While our transformation to achieve selective-failure blindness does not preserve uniqueness, it can still be combined with a modified version of their protocol. Hence, we can derive such oblivious transfer protocols based on unique blind signature schemes only (in the random oracle model), without necessarily requiring selective-failure blindness from scratch.

1 Introduction

Blind signatures, proposed by Chaum [Cha83], allow a signer to interactively sign messages for users such that the messages are hidden from the signer. Since their introduction many blind signatures schemes have been proposed [Cha83,JLO97,PS00,Abe01,Bol03,CKW04,KZ05,Fis06,Oka06,KZ08], and they typically share two basic security properties: *blindness* says that a malicious signer cannot

decide upon the order in which two messages have been signed in two executions with an honest user, and *unforgeability* demands that no adversarial user can create more signatures than interactions with the honest signer took place.

The security requirements for blind signatures have been formalized by Juels et al. [JLO97] and by Pointcheval and Stern [PS00]. Although these widely used definitions give basic security guarantees, blindness only holds in a restricted sense when it comes to aborted executions. That is, prior work does not guarantee blindness in case the malicious signer is able to learn if the user fails to compute a valid signature after the issuing protocol has concluded (i.e., in the post-processing step). However, in e-cash scenarios, for example, an honest user, unable to eventually derive a valid coin, will most likely complain to the malicious bank afterwards. Such aborts, where the malicious signer is informed afterwards which execution has failed (if any), have been recently considered by Camenisch et al. [CNS07]. As another example they consider a voting protocol based on blind signatures [FOO93, CNS07], where a malicious administrator can potentially deduce information about votes (possibly also for non-aborted executions) by causing some voters to abort and consulting the subsequent complaints.

As for user aborts and unforgeability, albeit the definitions [JLO97] and [PS00] are identical in spirit, the "one-more" notion in [PS00] leaves two possible interpretations: either the adversarial user is deemed to generate one more signature than executions with the signer have been *initiated* (i.e., even counting executions in which the user aborts), or the malicious user needs to output one more signature than executions have been *completed* (i.e., allowing "free" user aborts). In fact, this ambiguity re-appears in many works about blind signatures, some explicitly counting initiated executions [Bol03, Fis06, HKKL07], some emphatically referring to completed executions [JLO97, CKW04, Oka06, KZ08] and some remaining vague, too [Abe01, CNS07, HK07].

For both cases of aborts the stronger notions are desirable of course. For a blind signature scheme used to sign coins in an e-cash system, for instance, a malicious signer may otherwise abort executions deliberately and, by this, may be able to revoke unlinkability of coins. Vice versa, if unforgeability says that no adversarial user is able to create more signatures than interactions with the signer have been *initiated*, and no requirement about aborted sessions is imposed, then an adversarial user could potentially derive more signatures from such aborted executions. The signing bank could generally charge users for executions, which have stopped early. Yet, if the connection in the signing process breaks down accidently, the honest user is most likely unable to derive the coin and would hence be reluctant to pay for the transaction The bank may then gracefully waive the fee for such aborted executions, but still needs to handle forgery attempts.

Related Work. As mentioned before, Camenisch et al. [CNS07] have already considered the limitations of the standard blindness notion. They have introduced an extension called *selective-failure blindness* in which the a malicious signer should not be able to force an honest user to abort the signature issue protocol because of a certain property of the user's message, which would disclose some information about the message to the signer. They present a construction of a simulatable oblivious transfer protocols from so-called unique selective-failure blind signature schemes (in the random oracle model) for which the signature is uniquely determined by the message. Since the main result of the work [CNS07] is the construction of oblivious transfer protocols, the authors note that Chaum's scheme [Cha83] and Boldyreva's protocol [Bol03] are examples of such selective-failure blind schemes, but do not fully explore the relationship to (regular) blindness.

Hazay et al. [HKKL07] present a concurrently-secure blind signature scheme and, as part of this, they also introduce a notion called a-posteriori blindness. This notion considers blindness of

multiple executions between the signer and the user (as opposed to two sessions as in the basic case), and addresses the question how to deal with executions in which the user cannot derive a signature. However, the definition of a-posteriori blindness is neither known to be implied by ordinary blindness, nor implies it ordinary blindness (as sketched in [HKKL07]). Thus, selective-failure blindness does not follow from this notion.

Aborts of players have also been studied under the notion of fairness in two-party and multi-party computations, especially for the exchange of signatures, e.g. [Gol04, ASW98, GMPY06]. Fairness should guarantee that one party obtains the output of the joint computation if and only if the other party receives it. Note, however, that in case of blind signatures the protocol only provides a one-sided output to the user (namely, the signature). In addition, solutions providing fairness usually require extra assumptions like a trusted third party in case of disputes, or they add a significant overhead to the underlying protocol.

Our Results. We pick up the idea of selective-failure blindness to deal with signer aborts and expand the work of Camenisch et al. [CNS07] towards its relationship to blindness and further constructions of such schemes. We first show that selective-failure blindness is indeed a strictly stronger notion than regular blindness. We also extend the notion of selective-failure blindness to multiple executions, particularly addressing aborts of a subset of executions. We give two possible definitions for the multi-execution case and prove them to be equivalent. We then show that blindness in the basic case of two executions suffices to guarantee security in the case of many sessions and discuss the relation to a-posteriori blindness [HKKL07].

Next we present a general transformation which turns every secure blind signature scheme into a selective-failure blind scheme. Our transformation only requires an additional commitment of the message, which the user computes before the actual protocol starts and which the user then uses in the original protocol instead of the message itself.[1] Since the commitment is non-interactive, our transformation inherits important characteristics of the underlying protocol like the number of moves and concurrent security.

It should be noted, though, that the transformation destroys uniqueness (i.e., that each message has only one valid signature per key pair), as as required by [CNS07] to derive oblivious transfer from such blind signatures. However, we show that our transformation is still applicable if we modify the oblivious transfer protocol of [CNS07] slightly. Hence, we can now easily obtain an adaptive oblivious transfer from any unique blind signature scheme such that the protocol is simulatable in presence of failures. Put differently, we show that selective-failure blindness is not necessary to obtain such oblivious transfer protocols, but uniqueness is sufficient. We note that like the original protocol in [CNS07] this result is in the random oracle model.

We finally study the case of user aborts and show that every three-move blind signature scheme is unforgeable under user aborts. While this is clear for two-move schemes like Chaum's protocol [Cha83] our result shows that this remains true for other schemes like the ones by Pointcheval and Stern [PS00]. We show that, in general, this does not hold for schemes with four or more moves, assuming the existence of a secure two-move blind signature scheme. It remains open if there is a non-trivial and efficient transformation to take care of user aborts for schemes with more than three moves.[2]

[1] This idea has been conjectured by Hazay et al. [HKKL07] to also work for a-posteriori blindness. We are not aware of any formal claim or proof in the literature that using a commitment indeed provides security against aborts.

[2] By trivial transformations we refer for instance to a solution which ignores the underlying scheme and simply runs, say, Chaum's protocol.

In summary, our transformation to achieve selective-failure blindness, together with the result about user aborts, shows that any scheme with two or three moves can be efficiently turned into one, which is secure under aborts (of either party).

2 Blind Signatures

To define blind signatures formally we introduce the following notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ we denote the joint execution of $\mathcal{X}$ and $\mathcal{Y}$, where x is the private input of $\mathcal{X}$ and y defines the private input of $\mathcal{Y}$. The private output of $\mathcal{X}$ equals a and the private output of $\mathcal{Y}$ is b. We write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle^{\infty}}(y)$ if $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$ in arbitrarily interleaved order. Accordingly, $\mathcal{X}^{\langle \cdot, \mathcal{Y}(y_0) \rangle^1, \langle \cdot, \mathcal{Y}(y_1) \rangle^1}(x)$ can invoke arbitrarily ordered executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$, but interact with each algorithm only once.

Definition 2.1 (Blind Signature Scheme) *A blind signature scheme consists of a tuple of efficient algorithms* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *where*

Key Generation. $\mathsf{KG}_{\mathsf{BS}}(1^n)$ *for parameter* n *generates a key pair* $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}})$.

Signature Issuing. *The joint execution of algorithm* $\mathcal{S}(sk_{\mathsf{BS}})$ *and algorithm* $\mathcal{U}(pk_{\mathsf{BS}}, m)$ *for message* $m \in \{0,1\}^n$ *generates an output* σ *of the user (and some possibly empty output* λ *for the signer),* $(\lambda, \sigma) \leftarrow \langle \mathcal{S}(sk_{\mathsf{BS}}), \mathcal{U}(pk_{\mathsf{BS}}, m) \rangle$.

Verification. $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, \sigma)$ *outputs a bit.*

It is assumed that the scheme is complete*, i.e., for any* $n \in \mathbb{N}$*, any* $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$*, any message* $m \in \{0,1\}^n$ *and any* σ *output by* $\mathcal{U}$ *in the joint execution of* $\mathcal{S}(sk_{\mathsf{BS}})$ *and* $\mathcal{U}(pk_{\mathsf{BS}}, m)$ *we have* $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, \sigma) = 1$.

Security of blind signature schemes is defined by unforgeability and blindness [JLO97, PS00]. An adversary $\mathcal{U}^*$ against unforgeability tries to generate $k+1$ valid message-signatures pairs after at most k completed interactions with the honest signer, where the number of executions is adaptively determined by $\mathcal{U}^*$ during the attack. To identify completed sessions we assume that the honest signer returns a special symbol ok when having sent the final protocol message in order to indicate a completed execution (from its point of view). We remark that this output is "atomically" connected to the final transmission to the user.

The blindness condition says that it should be infeasible for a malicious signer $\mathcal{S}^*$ to decide which of two messages m_0 and m_1 has been signed first in two executions with an honest user $\mathcal{U}$. If one of these executions has returned $\perp$ then the signer is not informed about the other signature either.

Definition 2.2 (Secure Blind Signature Scheme) *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* secure *if the following holds:*

Unforgeability. *For any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{Unforge}^{\mathsf{BS}}_{\mathcal{U}^*}(n)$ *evaluates to* 1 *is negligible (as a function of* n*) where*

Experiment $\mathsf{Unforge}^{\mathsf{BS}}_{\mathcal{U}^*}(n)$
$(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$
$((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk_{\mathsf{BS}}), \cdot \rangle^\infty}(pk_{\mathsf{BS}})$
Return 1 *iff*
$m_i \neq m_j$ *for* $1 \leq i < j \leq k+1$, *and*
$\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m_i, \sigma_i) = 1$ *for all* $i = 1, 2, \ldots, k+1$, *and*
$\mathcal{S}$ *has returned* ***ok*** *in at most* k *interactions.*

Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(working in modes* ***find****,* ***issue*** *and* ***guess****) the probability that the following experiment* $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$ *evaluates to* 1 *is negligibly close to* $1/2$*, where*

Experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$
$(pk_{\mathsf{BS}}, m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{S}^*(\mathsf{find}, 1^n)$
$b \leftarrow \{0, 1\}$
$\mathsf{st}_{\mathsf{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_b)\rangle^1, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})\rangle^1}(\mathsf{issue}, \mathsf{st}_{\mathsf{find}})$
and let σ_b, σ_{1-b} *denote the (possibly undefined) local outputs of* $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$ *resp.* $\mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})$.
set $(\sigma_0, \sigma_1) = (\bot, \bot)$ *if* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$
$b^* \leftarrow \mathcal{S}^*(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{st}_{\mathsf{issue}})$
return 1 *iff* $b = b^*$.

3 Selective-Failure Blindness

In this section we review the definition of selective-failure blindness and show that selective-failure blindness is a strictly stronger requirement than the basic blindness property. Second, we discuss how to extend selective-failure blindness to multiple executions.

3.1 Definition

Camenisch et al. [CNS07] put forward the notion of *selective-failure blindness*, which says that a malicious signer $\mathcal{S}^*$ cannot force the user algorithm $\mathcal{U}$ to abort based on the specific message and that blindness should also hold in case the signer is able to learn that some executions have aborted. This is formalized by informing $\mathcal{S}^*$ which instance has aborted (i.e., if the left, the right, or both user instances have failed):

Definition 3.1 *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* selective-failure blind *if it is unforgeable (as in Definition 2.2) and the following holds:*

Selective-Failure Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(which works in modes* ***find****,* ***issue*** *and* ***guess****) the probability that experiment* $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$ *evaluates to* 1 *is negligibly close to* $1/2$ *where*

Experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$
$(pk_{\mathsf{BS}}, m_0, m_1, \beta_{\mathsf{find}}) \leftarrow \mathcal{S}^*(\mathsf{find}, 1^n)$
$b \leftarrow \{0, 1\}$
$\beta_{\mathsf{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_b)\rangle^1, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})\rangle^1}(\mathsf{issue}, \beta_{\mathsf{find}})$

and let σ_b, σ_{1-b} *denote the (possibly undefined) local outputs of* $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$ *resp.* $\mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})$.
define answer *as:* left *if only the first execution has failed,*
right *if only the second execution has failed,*
both *if both executions have failed,*
and (σ_b, σ_{1-b}) *otherwise.*
$b^* \leftarrow \mathcal{S}^*(\mathit{guess}, \mathsf{answer}, \beta_{\mathit{issue}})$
Return 1 *iff* $b = b^*$.

3.2 Relation to Regular Blindness

We first prove formally the fact that selective-failure blindness implies regular blindness. Then we separate the notion by turning a secure blind signature scheme into a one which is still secure but provably not selective-failure blind.

Proposition 3.2 *Every selective-failure blind signature scheme* $\mathsf{BS}_{\mathsf{SF}}$ *is also a secure blind signature scheme.*

Proof. Assume that there exists an adversarial controlled signer $\mathcal{A}$ breaking blindness with noticeable probability. Then we construct an attacker $\mathcal{S}^*$ breaking selective-failure blindness (with noticeable probability). The adversary $\mathcal{S}^*$ invokes a black-box simulation of $\mathcal{A}$. Whenever $\mathcal{A}$ interacts with the user algorithm (as described in the experiment), $\mathcal{S}^*$ forwards the messages (in both directions). At the end of the protocol $\mathcal{S}^*$ is informed if and which of the protocol executions have failed. In case that at least one of the user instances has aborted, the adversary $\mathcal{S}^*$ forwards the pair $(\perp, \perp)$ to $\mathcal{A}$, and otherwise, $\mathcal{S}^*$ obtains two signatures and hands them to $\mathcal{A}$. In both cases, $\mathcal{A}$ replies with a bit b^*, which $\mathcal{S}^*$ too outputs and stops.

A straightforward analysis shows that the success probabilities of $\mathcal{S}^*$ and $\mathcal{A}$ in the corresponding experiment are identical. Moreover, the notions of unforgeability are the same in both definitions. □

Proposition 3.3 *If there exists a secure blind signature scheme* BS*, then there exists a secure blind signature scheme* $\mathsf{BS}_{\overline{\mathsf{SF}}}$ *which is* not *selective-failure blind.*

Proof. We modify BS slightly into a scheme $\mathsf{BS}_{\overline{\mathsf{SF}}}$ which is identical to BS, except that we modify the key generation algorithm and add a break condition into the user algorithm. More precisely, let $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ be a secure blind signature scheme. We define the new blind signature scheme $\mathsf{BS}_{\overline{\mathsf{SF}}}$ as

KeyGen. $\mathsf{KG}_{\overline{\mathsf{SF}}}$ first sets $m_{\max} = 1^n$ as the maximum of the lexicographical order over n-bit strings. It then executes the key generation algorithm of the underlying blind signature scheme $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$ and returns $(sk_{\overline{\mathsf{SF}}}, pk_{\overline{\mathsf{SF}}}) = (sk_{\mathsf{BS}}, (pk_{\mathsf{BS}}, m_{\max}))$.

Signing Protocol. The interactive signing protocol remains unchanged except for one modification. The user algorithm checks *after* the last move of the protocol (and after computing the signature σ) that $m \leq m_{\max}$ and, if so, returns the signature σ, and $\perp$ otherwise.

Verification. The verification algorithm returns the result of $\mathsf{Vf}_{\mathsf{BS}}$.

The modified scheme is clearly complete, as the case $m > m_{\max}$ for an honest signer never occurs and because the initial protocol is complete. Obviously, if the blind signature scheme BS is unforgeable, then $\mathsf{BS}_{\overline{\mathsf{SF}}}$ is also unforgeable. This is easy to see as the malicious user may simply ignore the break condition.

Concerning blindness, first note that the malicious signer $\mathcal{S}^*$ is allowed to choose the public key and thus to pick some other value $m^*_{\max}$. As a malicious signer $\mathcal{S}^*$ is not informed which of the executions has failed (if any), setting some other value $m^*_{\max}$ than the predetermined maximum and possibly causing an abort does not lend any additional power to $\mathcal{S}^*$. To see this, note that the user algorithm does not abort prematurely if $m > m_{\max}$. Hence, from the (malicious) signer's point of view, the interaction is indistinguishable from an honest execution. It therefore follows that $\mathsf{BS}_{\overline{\mathsf{SF}}}$ still satisfies blindness.

We finally show that the modified scheme does not fulfill selective-failure blindness. Consider an malicious signer $\mathcal{S}^*$ in experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$. In the first step the adversary $\mathcal{S}^*$ computes a key pair $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$, it sets $m^*_{\max} = 10^{n-1}$ and picks two messages $m_0 = 0^n, m_1 = 1^n$ such that $m_0 \leq m^*_{\max} < m_1$. It outputs a public key $pk_{\overline{\mathsf{SF}}} = (pk_{\mathsf{BS}}, m^*_{\max})$ together with the message m_0, m_1 as defined in the first step of the experiment. Next, $\mathcal{S}^*$ has black-box access to two honest user instances (as described in experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$) where the first algorithm takes as input $(pk_{\overline{\mathsf{SF}}}, m_b)$ and the second user algorithm receives $(pk_{\overline{\mathsf{SF}}}, m_{1-b})$. In both executions $\mathcal{S}^*$ acts like the honest signer with key $sk_{\overline{\mathsf{SF}}} = sk_{\mathsf{BS}}$. Then $\mathcal{S}^*$ is eventually informed which of the executions has failed, i.e., receives left or right (as $\mathcal{S}^*$ has access to honest user instances, the case where both executions fail cannot occur by the completeness condition). The adversary $\mathcal{S}^*$ returns $b^* = 1$ if the left instance has failed, otherwise it returns $b^* = 0$.

It follows straightforwardly that the adversary $\mathcal{S}^*$ succeeds in predicting b with probability 1. □

3.3 Selective-Failure Blindness for Multiple Executions

The presumably natural way to extend selective-failure blindness to an arbitrary number of executions with user instances would be as follows. The malicious signer chooses q messages as well as a public key pk_{BS} and interacts with q user instances. We denote by π be a random permutation over $\{1, 2, \ldots, q\}$. The i-th user instance is initiated with the message $m_{\pi(i)}$ and the public key pk_{BS}. If at least one of the user instances aborts, then the adversary is given a binary vector v of length q indicating which of the user algorithms aborted. In the case that each execution allows the user to create a valid signature, then the adversary is given all message-signature pairs in non-permuted order.

In the final step the adversary tries to link a message-signature pair to an execution. There are two possible venues to formalize this. The first one, which we believe reflects much better the idea that the adversary should not be able to determine the order of signed messages, is to ask the adversary two output two indices i_0, i_1 such that $\pi(i_0) < \pi(i_1)$. The second version would be to ask the adversary to predict the connection much more explicitly, demanding to output indices (i, j) such that $\pi(i) = j$. Note that for the case of two executions both notions are equivalent.

Here we give the "order-based" definition and show in Appendix A that the two definitions are equivalent, assuming the following strengthening: During the signature issuing and in the final processing phase we give the malicious signer access to an oracle Reveal which for input i returns $\pi(i)$ and the user's signature σ_i if the execution has already finished successfully. This corresponds to the case that some coins in e-cash systems may have been spent meanwhile. Note that the reveal

oracle takes as input a state $\mathsf{st}^{\mathsf{rev}}$ where each signature is stored. The adversary's final choice i_0, i_1 must not have been disclosed, of course.

Definition 3.4 *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* multi-execution selective-failure blind *if it is unforgeable (as in Definition 2.2) and the following holds:*

Multi-Execution SF-Blindness. *For any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find, issue, *and* reveal*) the probability that experiment* $\mathsf{MSFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$ *returns* 1 *is negligibly close to* $\frac{1}{2}$*, where*

Experiment $\mathsf{MSFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$
$(pk_{\mathsf{BS}}, M, \beta_{\mathit{find}}) \leftarrow \mathcal{S}^*(\mathit{find}, 1^n)$ *where* $M = (m_1, \ldots, m_q)$ *with* $m_i \in \{0,1\}^n$
Select a random permutation π *over* $\{1, 2, \ldots, q\}$
$\beta_{\mathit{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}) \rangle^1, \ldots, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)}) \rangle^1, \mathsf{Reveal}(\cdot, \pi, \mathit{st}^{\mathit{rev}})}(\mathit{issue}, \beta_{\mathit{find}})$
and let $\sigma_{\pi(1)}, \ldots, \sigma_{\pi(q)}$ *denote the (possibly undefined) local outputs of* $\mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}), \ldots, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)})$*, immediately stored in* $\mathit{st}^{\mathit{rev}}$ *once an execution finishes (*$\mathit{st}^{\mathit{rev}}$ *is initially set to* $(\perp, \ldots, \perp)$*);* $\mathsf{Reveal}(\cdot, \pi, \mathit{st}^{\mathit{rev}})$ *is an oracle, which on input* i *returns* $(\pi(i), \mathit{st}^{\mathit{rev}}_i)$*.*
Return to $\mathcal{S}^*$ *all signatures* $v = (\sigma_1, \ldots, \sigma_q)$ *iff all executions have yielded valid signatures; otherwise return a vector* $v \in \{0,1\}^q$ *where the* i*-th entry is* 1 *if the* i*-th signature is valid, and* 0 *otherwise.*
$(i_0, i_1) \leftarrow \mathcal{S}^{*, \mathsf{Reveal}(\cdot, \pi, \mathit{st}^{\mathit{rev}})}(\mathit{reveal}, v, \beta_{\mathit{issue}})$
Return 1 *iff* $\pi(i_0) < \pi(i_1)$ *and* $\mathcal{S}^*$ *has never queried* Reveal *about* i_0, i_1*.*

The definition of multi-execution selective-failure blindness for the case $q = 2$ covers the standard definition of blindness. An adversary $\mathcal{A}$ breaking blindness can be used to build an adversary $\mathcal{S}^*$ breaking multi-execution selective-failure blindness as follows. The malicious signer $\mathcal{S}^*$ executes $\mathcal{A}$ in a black-box way and follows the blindness experiment until $\mathcal{S}^*$ receives either the signatures σ_0, σ_1 or the vector v. In case these two valid signatures are given to $\mathcal{S}^*$, it forwards both pairs to $\mathcal{A}$ and otherwise it outputs $\perp$. Finally, $\mathcal{S}^*$ outputs the decision bit b' returned by $\mathcal{A}$. The definition of selective-failure blindness is (semantically) identical to the definition of multi-execution selective failure blindness for the case $q = 2$.

Proposition 3.5 *A selective-failure blind signature scheme is also multi-execution selective-failure blind.*

Proof. Assume towards contradiction that BS is not multi-execution selective-failure blind. Then we show how to break selective-failure blindness for the case of two executions. The idea is as follows. The adversary $\mathcal{S}^*$ against selective-failure blindness executes a black-box simulation of the adversary $\mathcal{A}$ against multi-execution selective-failure blindness and tries to guess two executions for which $\mathcal{A}$ tries to succeed and does not make reveal queries (which is necessary for $\mathcal{A}$ to win). Adversary $\mathcal{S}^*$ inserts the data from its two sessions with honest users in the corresponding executions in the simulation and plays the honest users in the other executions itself. Then it uses $\mathcal{A}$'s final output to guess the order of the two executions.

More precisely, let $\mathcal{A}$ be an adversary breaking multi-execution selective-failure blindness with probability noticeably greater than $\frac{1}{2}$.Algorithm $\mathcal{S}^*$ against selective-failure blindness for two executions runs a black-box simulation of $\mathcal{A}$ which initially outputs a pair (pk_{BS}, M) consisting of

a public key pk_{BS} and a vector $M = (m_1, \ldots, m_q)$ of q messages. Adversary $\mathcal{S}^*$ selects random indices i'_0, i'_1 between 1 and q as well as a random permutation π over $\{1, 2, \ldots, q\}$. It outputs the triple $(pk_{\mathsf{BS}}, m_{\pi(i'_0)}, m_{\pi(i'_1)})$ (and a state).

In the next part algorithm $\mathcal{S}^*$ has access to two (external) user instances. It also initializes $q-2$ honest user instances $\mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(i)})$ for $i \neq i'_0, i'_1$ and continues the simulation of $\mathcal{A}$. For each user instance $i \neq i'_0, i'_1$ (with which adversary $\mathcal{A}$ communicates with) adversary $\mathcal{S}^*$ acts on behalf of the honest user. For $i = i'_0$ or $i = i'_1$ algorithm $\mathcal{S}^*$ relays the communication between $\mathcal{A}$ and the two external user instances.

Algorithm $\mathcal{S}^*$ eventually receives the information that one of the two executions has failed (or both), or receives two valid signatures. If all q user algorithms have returned valid signatures, then $\mathcal{S}^*$ forwards the q messages-signature pairs to $\mathcal{A}$. Otherwise, it computes the vector v of length q indicating which of the executions aborted (note that for $i \neq i'_0, i'_1$ algorithm $\mathcal{S}^*$ can easily determine the status as it runs the user copy itself).

During the experiment, algorithm $\mathcal{A}$ is allowed to invoke its reveal oracle at most $q-2$ times. The adversary $\mathcal{S}^*$ answers each query $i \neq i'_0, i'_1$ by sending the pair $(\pi(i), \sigma_i)$ (where σ_i may be undefined). In the case that $\mathcal{A}$ tries to reveal an execution which took place with an external user algorithm, i.e., queries about i'_0 or i'_1, then $\mathcal{S}^*$ immediately stops and outputs a random bit b'. Otherwise, if the algorithm $\mathcal{A}$ finally outputs two indices i_0, i_1 equal to i'_0 and i'_1, then $\mathcal{S}^*$ outputs $b' = 0$ if $i_0 = i'_0$ and $b' = 1$ if $i_0 = i'_1$. In the case that $\mathcal{A}$ outputs different indices, then $\mathcal{S}^*$ also returns a random bit b'.

For the analysis of $\mathcal{S}^*$ recall that algorithm $\mathcal{A}$ outputs two values i_0, i_1 such that $\pi(i_0) < \pi(i_1)$ with probability $\delta(n) \geq \frac{1}{2} + \epsilon(n)$ where $\epsilon(n)$ is noticeable. The guess i'_0, i'_1 of algorithm $\mathcal{S}^*$ is correct with probability $1/q^2$ —noting that the simulation is perfect up to the point where a bad reveal query is made or till the adversary stops— and in this case we have $b = 0$ iff $i_0 = i'_0$ and $b = 1$ iff $i_0 = i'_1$. In any other case $\mathcal{S}^*$ returns a random bit and succeeds with probability $\frac{1}{2}$. Altogether $\mathcal{S}^*$ succeeds with probability

$$\frac{1}{q^2} \cdot \left(\frac{1}{2} + \epsilon(n)\right) + \frac{1}{2} \cdot \left(1 - \frac{1}{q^2}\right) = \frac{1}{2} + \frac{\epsilon(n)}{q^2},$$

which is larger than $\frac{1}{2}$ by a noticeable amount. □

3.4 Relation to A-Posteriori Blindness

In the following we discuss the relation between selective-failure blindness and a-posteriori blindness [HKKL07]. Roughly speaking, a-posteriori blindness advocates that blindness only needs to hold for non-aborted sessions. Hazay et al. formalize this basic idea in an experiment where the adversary first outputs a public key pk together with a message distribution $\mathcal{M}$. The malicious signer then concurrently interacts with ℓ honest user instances, where each user instance gets as input the public key pk and a message sampled according to $\mathcal{M}$. Afterwards, when the signer has finished all ℓ interactions, it receives ℓ' message-signature pairs in a randomly permuted order, where $1 \leq \ell' \leq \ell$ denotes the number of non-aborted executions. The adversary wins the game if it associates one non-aborted execution to a messages-signature pair. A detailed discussion about a-posteriori blindness in the concurrent setting is given in [HKKL07].

From a syntactically point of view there are numerous differences between the definition of selective-failure blindness and a-posteriori blindness. Firstly, the adversary in our security definition picks the messages, whereas in the experiment of a-posteriori blindness it only chooses a

message distribution. Secondly, in contrast to a-posteriori blindness, the malicious signer in our case receives the information which of the user instances have aborted. In an e-cash scenario, this corresponds to the case where a user (who may have completed all rounds of the protocol) could not derive a valid coin and informs the signing bank about this problem. Thirdly, we have chosen an ordering-based definition of multi-execution selective-failure blindness (Definition 3.4), where the adversary has to distinguish the order of two different executions. In contrast, the work by Hazay et al. [HKKL07] uses a a prediction-based definition where the malicious signer has to link an execution to a message-signature pair. Recall that, for our notion of selective-failure blindness, both definitions are equivalent (Appendix A).

Finally, the attacker in our definitions has access to a reveal oracle that discloses the message used during a specific execution. Such an oracle is also not considered in the definition of a-posteriori blindness. In the real world, this oracle represents side information the signer obtains, e.g., customer A opens up a bank account before customer B. Then customer B cannot withdraw coins before having opened up an account and every meanwhile spent coin has to be from customer A. Note that these side information provide the malicious signer also with information about the non-aborted executions. From a technical point of view, the reveal oracle allows us to prove the equivalence between selective-failure blindness for two executions and for multiple executions, as well as the equivalence of the two types of multi-execution selective-failure blindness definitions.

A natural question is whether the definition of a-posteriori blindness and the definition of multi-execution selective-failure blindness are equivalent for the special case of two executions. To answer this question we briefly recall the counter example of Hazay el al. which shows that a-posteriori blindness *does not* imply regular blindness. This example consists of a scheme that satisfies a-posteriori blindness but that easily violates blindness. In this scheme, the honest user algorithms validates the first received message from the signer. In the case that this message is improper, then it sends the message m to the signer and aborts afterwards. Since a-posteriori blindness only guarantees blindness for non-aborted sessions, this scheme remains a-posteriori blind. However, it follows easily that this scheme is not blind. Hence, a-posteriori blindness cannot be equivalent to selective-failure blindness, because selective-failure blindness does imply regular blindness.

4 From Blindness to Selective-Failure Blindness

In this section we show how to turn every secure blind signature scheme BS into a selective-failure blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$. The high-level idea is to modify BS slightly into $\mathsf{BS}_{\mathsf{SF}}$ by executing BS with a non-interactive commitment com of the message m (instead of the message itself).

Definition 4.1 (Commitment Scheme) *A (non-interactive) commitment scheme consists of a tuple of efficient algorithms* $\mathcal{C} = (\mathsf{KG}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Vf}_{\mathsf{com}})$ *where*

Key Generation. *Algorithm* $\mathsf{KG}_{\mathsf{com}}(1^n)$ *on input the security parameter outputs a key* pk_{com}.

Commitment Phase. *Algorithm* Com *takes as input* pk_{com} *as well as* $m \in \{0,1\}^n$ *and outputs* $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m)$.

Verification. $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com})$ *outputs a bit.*

It is assumed that the commitment scheme is complete*, i.e., for any* $n \in \mathbb{N}$*, any* $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$*, for any message* $m \in \{0,1\}^n$ *and any* $(\mathsf{decom}, \mathsf{com}) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m)$ *we have* $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com}) = 1$.

Security of commitment schemes is defined by secrecy and unambiguity. Secrecy guarantees that the receiver cannot learn the message from the commitment and unambiguity says that the sender cannot change the message anymore once the commitment phase is over. Here we use a slightly different way to define secrecy compared to the usual approach in the literature, but it is easy to see by a hybrid argument that our definition is equivalent:

Definition 4.2 (Secure Commitment) *A (non-interactive) commitment scheme* $\mathcal{C} = (\mathsf{KG}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Vf}_{\mathsf{com}})$ *is called secure if the following holds:*

Secrecy. *For any efficient algorithm* $\mathsf{R}^*_{\mathsf{com}}$ *(working in modes* ***find*** *and* ***guess****) the probability that experiment* $\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{com}}}(n)$ *evaluates to* 1 *is negligibly close to* 1/2*.*

Experiment $\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{com}}}(n)$
$(m_0, m_1, pk_{\mathsf{com}}, \beta_{\mathit{find}}) \leftarrow \mathsf{R}^*_{\mathsf{com}}(\mathit{find}, 1^n)$
$b \leftarrow \{0,1\}$
$\mathsf{com}_b \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_b)$ *and* $\mathsf{com}_{1-b} \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_{1-b})$
$b^* \leftarrow \mathsf{R}^*_{\mathsf{com}}(\mathit{guess}, \beta_{\mathit{find}}, \mathsf{com}_0, \mathsf{com}_1)$
Return 1 *iff* $b = b^*$*.*

Unambiguity. *For any efficient algorithm* $\mathsf{S}^*_{\mathsf{com}}$ *the probability that experiment* $\mathsf{Unambiguity}^{\mathcal{C}}_{\mathsf{S}^*_{\mathsf{com}}}(n)$ *evaluates to* 1 *is negligible.*

Experiment $\mathsf{Unambiguity}^{\mathcal{C}}_{\mathsf{S}^*_{\mathsf{com}}}(n)$
$pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$
$(m, m', \mathsf{decom}, \mathsf{decom}') \leftarrow \mathsf{S}^*_{\mathsf{com}}(pk_{\mathsf{com}})$
Return 1 *iff*
$\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{Com}) = 1$ *and*
$\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m', \mathsf{decom}', \mathsf{Com}) = 1$ *as well as* $m \neq m'$*.*

Note that such commitment schemes exist under standard assumptions like pseudorandom generators [Nao91] or hash functions [DPP97]. In order to use a commitment in a blind signature scheme —which we defined to take messages of n bits— we need that the commitment scheme is *length-invariant*, meaning that for n-bit messages the commitment itself is also n bits. This can always be achieved by using a collision-resistant hash function (with n bits output) on top.

Construction 4.3 (Selective-Failure Blind Signature Scheme $\mathsf{BS}_{\mathsf{SF}}$**)** *Let* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *be a blind signature scheme and* $\mathcal{C}$ *be a length-invariant commitment scheme. Define the blind signature scheme* $\mathsf{BS}_{\mathsf{SF}}$ *through the following three procedures:*

Key Generation. *The generation algorithm* $\mathsf{KG}_{\mathsf{SF}}(1^n)$ *executes the key generation algorithm of the blind signature scheme* BS*,* $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$*. It also runs the key generation algorithm for the commitment scheme,* $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$*. It returns the private key* $sk_{\mathsf{SF}} = sk_{\mathsf{BS}}$ *and the public key* $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$*.*

Signature Issue Protocol. *The interactive signature issue protocol for message* $m \in \{0,1\}^n$ *is described in Figure 1.*

Signature Verification. *The verification algorithm* $\mathsf{Vf}_{\mathsf{SF}}(pk_{\mathsf{SF}}, m, \sigma')$ *for* $\sigma' = (\sigma, \mathsf{decom}, \mathsf{com})$ *returns* 1 *iff* $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, \sigma, \mathsf{com}) = 1$ *and* $\mathsf{Vf}_{\mathsf{com}}(pk_{\mathsf{com}}, m, \mathsf{decom}, \mathsf{com}) = 1$*.*

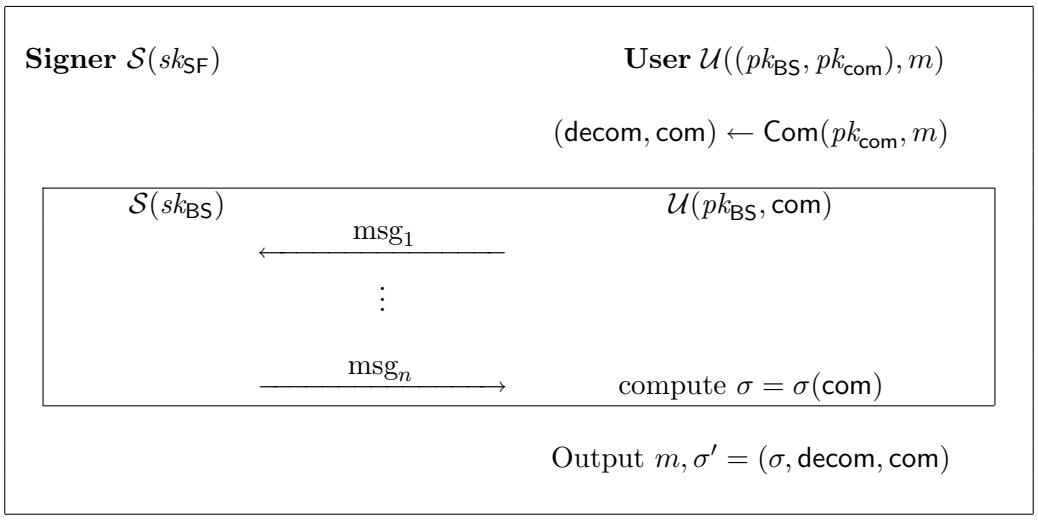

Figure 1: Issue protocol of the blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$

Theorem 4.4 *If* BS *is a secure blind signature scheme and* $\mathcal{C}$ *is a secure, length-invariant commitment scheme, then the scheme* $\mathsf{BS}_{\mathsf{SF}}$ *in Construction 4.3 is a selective-failure blind signature scheme.*

We note that, if the starting blind signature scheme provides statistical blindness, and the commitment scheme is also statistically-hiding, then the derived protocol achieves selective-failure blindness in a statistical sense. This can be seen from the proof of the theorem, which is split into two claims, covering unforgeability and selective-failure blindness:

Claim 1: $\mathsf{BS}_{\mathsf{SF}}$ is unforgeable.

In the proof we distinguish between two cases. The first case occurs if the adversary $\mathcal{U}^*$ succeeds in outputting $k+1$ valid pairs $m_i, \sigma'_i = (\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ such that the commitments com_i are pairwise different. But then we can break the unforgeability of the underlying blind signature scheme BS. In the second case $\mathcal{U}^*$ succeeds and at least two commitments $\mathsf{com}_i, \mathsf{com}_j$ (with $i \neq j$) are identical. But then we can break the unambiguity of the commitment scheme $\mathcal{C}$.

Proof. Assume to the contrary that the resulting selective-failure blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$ is *not* unforgeable. Then there exists an adversary $\mathcal{U}^*$ breaking unforgeability with noticeable probability, i.e., on input pk_{SF} the algorithm $\mathcal{U}^*$ returns $k+1$ valid signatures $\sigma'_i = (\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ for messages m_i after at most k interactions with the honest signer $\mathcal{S}$. Note that here we do not deal with user aborts and count any initiated interaction; the case of counting only completed interactions is taken care of in the next section.

We first take a look at the success probability of $\mathcal{U}^*$, we have

$$\psi(n) := \mathrm{Prob}\left[\mathsf{Forge}^{\mathsf{BS}_{\mathsf{SF}}}_{\mathcal{U}^*}(n) = 1\right]$$

where $\psi(n)$ is noticeable. This probability can be separated according to the two exclusive events that $\mathcal{U}^*$ succeeds and all commitments com_i are different, with the corresponding probability denoted by $\psi_0(n)$, and into the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds and at least two commitments are identical

(with probability $\psi_1(n)$) According to our assumption that $\psi(n)$ is noticeable, $\psi_0(n)$ or $\psi_1(n)$ (or both) must be noticeable.

We next construct out of $\mathcal{U}^*$ algorithms $\mathcal{A}_{\mathsf{UNF}}$ and $\mathcal{A}_{\mathsf{UNA}}$ against unforgeability of BS and unambiguity of the commitment scheme $\mathcal{C}$.

Attacking Unforgeability. The adversary $\mathcal{A}_{\mathsf{UNF}}$ takes as input the public key pk_{BS} of the blind signature scheme BS and works as follows. It executes the key generation algorithm of the commitment scheme $pk_{\mathsf{com}} \leftarrow \mathsf{KG}_{\mathsf{com}}(1^n)$ and runs a black-box simulation of $\mathcal{U}^*$ on input $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$. The signer instances in the attack of $\mathcal{U}^*$ are simulated with the help of the external signer instances accessible by $\mathcal{A}_{\mathsf{UNF}}$, i.e., adversary $\mathcal{A}_{\mathsf{UNF}}$ relays the communication between $\mathcal{U}^*$ and its signer instance oracle $\mathcal{S}(sk_{\mathsf{BS}})$ (as described in experiment $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}$). When $\mathcal{U}^*$ finishes its attack, it outputs $k+1$ message-signatures pairs m_i, σ_i' after at most k interactions. Now $\mathcal{A}_{\mathsf{UNF}}$ parses each σ_i' as $(\sigma_i, \mathsf{decom}_i, \mathsf{com}_i)$ and returns the $k+1$ pairs com_i, σ_i and stops.

Assume that $\psi_0(n)$, the probability that $\mathcal{U}^*$ succeeds and all com_i's are distinct, is noticeable. Then, since the simulation is perfect from the viewpoint of $\mathcal{U}^*$, adversary $\mathcal{A}_{\mathsf{UNF}}$ succeeds in outputting $k+1$ valid pairs com_i, σ_i for distinct "messages" com_i with noticeable probability, too, contradicting the unforgeability property of the underlying blind signature scheme. Note also that the numbers of initiated and completed executions are identical in both cases.

Attacking Unambiguity. In order to break the unambiguity of $\mathcal{C}$, the adversary $\mathcal{A}_{\mathsf{UNA}}$ takes as input the public key pk_{com} of the commitment scheme $\mathcal{C}$ and works as follows. It executes the key generation algorithm of the blind signature scheme $(sk_{\mathsf{BS}}, pk_{\mathsf{BS}}) \leftarrow \mathsf{KG}_{\mathsf{BS}}(1^n)$ as well as a the honest signer algorithms $\mathcal{S}(sk_{\mathsf{BS}})$ and runs a black-box simulation of $\mathcal{U}^*$ on input $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$. Note that running the program of the honest signer on input sk_{BS} simulates each execution with a signer instance. Algorithm $\mathcal{U}^*$ eventually returns $k+1$ message-signature pairs (m_i, σ_i') after at most k interactions with $\mathcal{S}$. The adversary $\mathcal{A}_{\mathsf{UNA}}$ then checks if there are valid signatures with $\mathsf{com}_i = \mathsf{com}_j$ for some $i \neq j$ and, if so, outputs two tuples $(m_i, \mathsf{decom}_i, \mathsf{com}_i), (m_j, \mathsf{decom}_j, \mathsf{com}_j)$ such that $m_i \neq m_j$ and $\mathsf{com}_i = \mathsf{com}_j$. If not, it outputs a failure message.

For the analysis note that the simulation again perfectly mimics the original attack of $\mathcal{U}^*$. Hence, if $\psi_1(n)$ is noticeable, then such $\mathsf{com}_i = \mathsf{com}_j$ with valid decommitments for $m_i \neq m_j$ appear with noticeable probability, and the commitment adversary $\mathcal{A}_{\mathsf{UNA}}$ therefore finds an ambiguous commitment with this probability, too. But this clearly violates the security of the commitment scheme $\mathcal{C}$. □

Claim 2: $\mathsf{BS}_{\mathsf{SF}}$ is selective-failure blind.

The high-level idea of the proof is as follows. We again distinguish between two cases. In the first case the adversary $\mathcal{A}_{\mathsf{SF}}$ succeeds with noticeable probability and both message-signature pairs are valid. But then we show how to break the blindness property of the underlying blind signature scheme BS. We next argue that in the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds with noticeable probability and forces at least one of the user algorithms to fail, then we are able to break the secrecy of the commitment scheme (because then the only information available to the signer are the commitments of the messages).

Proof. Assume towards contradiction that the resulting blind signature scheme $\mathsf{BS}_{\mathsf{SF}}$ is *not* selective-failure blind, and that there exists a successful adversary $\mathcal{A}_{\mathsf{SF}}$ against selective-failure blindness.

Let

$$\delta(n) := \text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1\right] = \tfrac{1}{2} + \epsilon(n)$$

where $\epsilon(n) = \delta(n) - \frac{1}{2}$ is noticeable. We divide the success case according to the two exclusive events that $\mathcal{A}_{\mathsf{SF}}$ succeeds and that both message-signature pairs are valid (event valid) and into the case where $\mathcal{A}_{\mathsf{SF}}$ succeeds and at least one of the signatures is not valid (event $\neg\mathsf{valid}$). Then,

$$\begin{aligned}\text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1\right] - \tfrac{1}{2} = \text{Prob}[\,\mathsf{valid}] \cdot \left(\text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \mathsf{valid}\right] - \tfrac{1}{2}\right) \\ + \text{Prob}[\,\neg\mathsf{valid}] \cdot \left(\text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \neg\mathsf{valid}\right] - \tfrac{1}{2}\right).\end{aligned}$$

According to our assumption that $\delta(n)$ is noticeable, either the first term, denoted $\delta_0(n)$, or the second term $\delta_1(n)$ has to be noticeable (or both are noticeable). We next turn $\mathcal{A}_{\mathsf{SF}}$ into algorithms $\mathcal{A}_{\mathsf{blind}}$ and $\mathcal{A}_{\mathsf{com}}$ against regular blindness and secrecy of the commitment scheme, respectively.

Attacking Blindness. The adversary $\mathcal{A}_{\mathsf{blind}}$ works as follows. It runs a black-box simulation of $\mathcal{A}_{\mathsf{SF}}$, which initially outputs two messages (m_0, m_1) together with a public key pk_{SF}. The adversary $\mathcal{A}_{\mathsf{blind}}$ extracts pk_{BS} and pk_{com} from pk_{SF} and calculates the commitments (and decommitments) $(\mathsf{decom}_0, \mathsf{com}_0) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_0)$ and $(\mathsf{decom}_1, \mathsf{com}_1) \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, m_1)$. It outputs $\mathsf{com}_0, \mathsf{com}_1$ and pk_{BS}. It is then given access to two user instances $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_b)$ and $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_{1-b})$ for a unknown bit b and relays the communication between these instances and $\mathcal{A}_{\mathsf{SF}}$. If, at the end, at least one of the (external) user algorithms fails, then $\mathcal{A}_{\mathsf{blind}}$ outputs a random bit and stops. Otherwise, it augments σ_0, σ_1 to $\sigma_0' = (\sigma_0, \mathsf{decom}_0, \mathsf{com}_0)$ and $\sigma_1' = (\sigma_1, \mathsf{decom}_1, \mathsf{com}_1)$ and returns the two signatures σ_0', σ_1' (obtained by the external user algorithms) to $\mathcal{A}_{\mathsf{SF}}$. The final output of $\mathcal{A}_{\mathsf{blind}}$ consists of the bit b^* returned by $\mathcal{A}_{\mathsf{SF}}$.

Note that $\mathcal{A}_{\mathsf{blind}}$ simulates the experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ by executing the blindness experiment for the underlying blind signature scheme BS and by computing the commitments internally. Hence, the case where both message-signature pairs are valid is the one where experiment $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ is identical to experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n)$. If one of the signatures is invalid, then $\mathcal{A}_{\mathsf{blind}}$ returns a random bit. Therefore, the success probability of $\mathcal{A}_{\mathsf{blind}}$ in experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n)$ can be calculated as:

$$\begin{aligned}&\text{Prob}\left[\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{blind}}}(n) = 1\right] \\ &\quad= \text{Prob}[\,b = b^* \wedge\ \neg\mathsf{valid}] + \text{Prob}[\,b = b^* \wedge\ \mathsf{valid}] \\ &\quad= \text{Prob}[\,b = b^*\ |\,\mathsf{valid}] \cdot \text{Prob}[\,\mathsf{valid}] + \text{Prob}[\,b = b^*\ |\,\neg\mathsf{valid}] \cdot \text{Prob}[\,\neg\mathsf{valid}]\,. \\ &\quad= \text{Prob}[\,\mathsf{valid}] \cdot \text{Prob}[\,b = b^*\ |\,\mathsf{valid}] + \tfrac{1}{2} \cdot (1 - \text{Prob}[\,\mathsf{valid}]) \\ &\quad= \text{Prob}[\,\mathsf{valid}] \cdot \text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \mathsf{valid}\right] + \tfrac{1}{2} \cdot (1 - \text{Prob}[\,\mathsf{valid}]) \\ &\quad= \tfrac{1}{2} + \text{Prob}[\,\mathsf{valid}] \cdot \left(\text{Prob}\left[\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n) = 1 \;\middle|\; \mathsf{valid}\right] - \tfrac{1}{2}\right) \\ &\quad= \tfrac{1}{2} + \delta_0(n).\end{aligned}$$

According to our assumption that $\delta_0(n)$ is noticeable it follows that $\mathcal{A}_{\mathsf{blind}}$ breaks the blindness of the underlying blind signature scheme BS with noticeable probability. This, however, contradicts our assumption that BS is a *secure* blind signature scheme.

Attacking Secrecy of the Commitment. In order to break the secrecy of the commitment scheme $\mathcal{C}$, the adversary $\mathcal{A}_{\mathsf{com}}$ executes a black-box simulation of $\mathcal{A}_{\mathsf{SF}}$, which initially outputs two messages (m_0, m_1) as well as a public key pk_{SF}. The adversary $\mathcal{A}_{\mathsf{com}}$ extracts the keys pk_{com} and pk_{BS} from pk_{SF} and outputs $(m_0, m_1, pk_{\mathsf{com}})$ for the secrecy experiment of the commitment scheme. It then receives two commitments $\mathsf{com}_0, \mathsf{com}_1$, one for message m_b and the other one for message m_{1-b} (without knowing which commitment corresponds to which message).

The adversary now runs (in the role of the honest user $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_0)$ and $\mathcal{U}(pk_{\mathsf{BS}}, \mathsf{com}_1)$) the selective-failure blindness experiment with $\mathcal{A}_{\mathsf{SF}}$. At the end of the issue protocol each user instance returns either a signature for the commitment or $\perp$. In the case that both user algorithms return a valid signature, then $\mathcal{A}_{\mathsf{com}}$ outputs a random bit b^* and stops. Otherwise, if both user algorithms have failed, then $\mathcal{A}_{\mathsf{com}}$ sends the value both to $\mathcal{A}_{\mathsf{SF}}$. In the case that the first user algorithm has failed, then $\mathcal{A}_{\mathsf{com}}$ returns left to $\mathcal{A}_{\mathsf{SF}}$ and else (if the second user algorithm has failed), it forwards right to $\mathcal{A}_{\mathsf{SF}}$. The final output of $\mathcal{A}_{\mathsf{com}}$ consists of the bit b^* returned by $\mathcal{A}_{\mathsf{SF}}$.

The adversary $\mathcal{A}_{\mathsf{com}}$ simulates the experiment of selective-failure blindness perfectly, up to the point where it obtains the (possibly undefined) signatures. Given that at least one of them is invalid, the simulation corresponds to the case $\mathsf{SFBlind}^{\mathsf{BS}}_{\mathcal{A}_{\mathsf{SF}}}(n)$ (given $\neg\mathsf{valid}$) for the same choice b as in the commitment experiment. Else, $\mathcal{A}_{\mathsf{com}}$ outputs a random bit. A simple calculation similar to the previous case now shows that

$$\mathrm{Prob}\left[\mathsf{Secrecy}^{\mathcal{C}}_{\mathsf{R}^*_{\mathsf{com}}}(n) = 1\right] = \tfrac{1}{2} + \delta_1(n).$$

If $\delta_1(n)$ is noticeable, it follows that $\mathcal{A}_{\mathsf{com}}$ breaks the secrecy of the commitment scheme with noticeable probability, contradicting the security of $\mathcal{C}$. □

5 Unforgeability and User Aborts

In this section we consider executions in which an adversarial controlled user may abort sessions and the unforgeability requirement with respect to *initiated* or *completed* executions with the signer. For sake of distinction we call the requirement where the adversary has to find $k+1$ valid message-signature pairs after k initiated executions *weak unforgeability*, and the originally given definition charging only completed executions *unforgeability under user aborts*.

We show in the following that every three-move blind signature scheme, which is weakly unforgeable is also unforgeable under user aborts. Note that in three-move schemes, for a meaningful protocol, the first message is always sent by the signer. As such we may think of two-move schemes as having an additional first move in which the signer simply sends an empty message (although the claim for two-move schemes follows straightforwardly anyway).

We remark that we leave the scheduling of transmissions fully up to the adversary controlling the users, i.e., the adversary decides when to send messages to the signer and when the signer's messages are delivered to the user. Only the signer's output ok is given immediately after the signer's final message has been delivered.

Theorem 5.1 *Every secure blind signature scheme with at most three moves is unforgeable under user aborts.*

The proof idea is that we can delay the delivery of the user's message in an execution till we can be sure that the adversary completes this execution. If the execution is not completed, then

we can simply disregard the original user message, finish the protocol ourselves as an honest user and create another valid signature in addition to the forgeries of the adversary.

Proof. Let us assume that BS is secure a blind signature scheme with (at most) three moves which is *not* unforgeable under user aborts, i.e., there exists an adversary $\mathcal{A}$ which outputs $k+1$ valid message-signature pairs with noticeable probability after at most k completed sessions (in which the signer has output ok). Then we show via an algorithm $\mathcal{U}^*$ (derived from $\mathcal{A}$) how to refute the weak unforgeability of BS (this time counting all initiated sessions).

Algorithm $\mathcal{U}^*$ runs a black-box simulation of $\mathcal{A}$, mainly relaying the data between the external signer instance and $\mathcal{A}$. Each time the adversary $\mathcal{A}$ initiates a new protocol instance with the signer, $\mathcal{U}^*$ sends the first message received from the honest signer $\mathcal{S}$ to $\mathcal{A}$. In the case that $\mathcal{A}$ asks to deliver the second message of this protocol, $\mathcal{U}^*$ saves it until $\mathcal{A}$ demands to deliver the third message. We call this execution *open*. If the request for the signer's reply eventually comes then $\mathcal{U}^*$ first delivers the previously stored second message to the external signer instance, receives the answer and only then delivers it to $\mathcal{A}$. The session becomes *closed*.

At the end of $\mathcal{A}$'s simulation, when $\mathcal{A}$ stops and outputs the $k+1$ message-signature pairs $((m_1,\sigma_1),\ldots,(m_{k+1},\sigma_{k+1}))$ there may still be open executions. For each such execution $\mathcal{U}^*$ disregards the stored message, searches (in lexicographical order) the first "fresh" $m \in \{0,1\}^n$ such that m is different from the messages in the closed or meanwhile terminated executions, and completes the session by running the honest user $\mathcal{U}(pk_{\mathsf{BS}}, m)$ for the remaining steps (note that this is possible as the user has not sent anything yet). Then $\mathcal{U}^*$ outputs the $k+1$ message-signature pairs returned by $\mathcal{A}$ plus all the additional pairs created by terminating open sessions.

For the analysis recall that $\mathcal{A}$ returns $k+1$ valid message-signature pairs after at most k terminated protocol executions. Suppose the adversary $\mathcal{A}$ has aborted ℓ protocol instances (open executions), which $\mathcal{U}^*$ has finally finished. By the completeness of the scheme $\mathcal{U}^*$ obtained a valid message-signature pair in all of these ℓ executions, and thus all $k+\ell+1$ pairs verify correctly. In addition, the other messages in the open executions are picked such that they differ from the $k+1$ messages returned by $\mathcal{A}$ and from each other (note that all these messages can be found easily in polynomial time). Altogether, $\mathcal{U}^*$ creates more valid signatures for different messages than executions with the signer have been started (namely, $k+\ell$ initiated executions). □

We note that the result above is optimal in the sense that for four or more moves no such claim can be made (if there are secure schemes with two moves):

Proposition 5.2 *Every secure blind signature scheme* BS *with two moves can be converted into a secure blind signature scheme* $\mathsf{BS}_{\overline{\mathsf{UuA}}}$ *with four moves, which is weakly unforgeable but* not *unforgeable under user aborts.*

Proof. Let BS be a secure blind signature scheme with two-moves. We modify BS to a blind signature scheme $\mathsf{BS}_{\overline{\mathsf{UuA}}}$ which is identical to BS, except that we add two additional moves at the end, where the user simply transmits the bit 0 and the signer also replies with the bit 0.

Obviously, the modification neither affects weak unforgeability nor blindness. Nevertheless, the modified blind signature scheme is *not* unforgeable under user aborts. A malicious user $\mathcal{U}^*$ can easily abort an interaction after the second move, allowing him to compute the signature of the underlying blind signature scheme, but such that the signer does not output ok. In other words, $\mathcal{U}^*$ can create signatures without ever completing a single session. □

The previous proposition does not rule out that there is a *transformation* turning schemes with four or more moves into unforgeable ones under user aborts. An apparent approach is to ignore the original protocol and two run a scheme, which already has this property (like Chaum's two-move blind signature scheme in the random oracle model). Yet, it is preferable of course to have a lightweight transformation adhering to the basics of the underlying protocol (like the avoidance of random oracles or general but expensive multi-party protocols).

6 Selective Failures and Adaptive Oblivious Transfer

Camenisch et al. [CNS07] also show how to construct an adaptive oblivious transfer protocol out of any unique selective-failure blind signature scheme (in the random oracle model). Roughly speaking, uniqueness means that each message has only one signature per public key. More formally, a blind signature scheme is *unique* [GO92, CNS07] if for every (possibly maliciously chosen) public key pk_{BS} and every message $m \in \{0,1\}^*$, there exists at most one signature $s \in \{0,1\}^*$ such that $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, m, s) = 1$.

In this section we focus on the question whether our transformation turning every blind signature into one with selective-failure blindness is applicable. We have already mentioned in the introduction that the initial commitment destroys uniqueness of the blind signature scheme because each message may have several valid signatures per key pair. Here we show that is nonetheless possible to build an adaptive k-out-of-N oblivious transfer protocol out of *any* unique blind signature scheme by applying our transformation. The following construction is a modification of the protocol in [CNS07] and, because of the problems with uniqueness, we have to prove the security of this construction from scratch, digging also into the proof of selective-failure blindness for our transformation.

6.1 Simulatable Adaptive Oblivious Transfer

Oblivious Transfer (OT), proposed by Rabin [Rab81], is an interactive protocol between a sender S and a receiver R. The sender in this protocol gets as input N messages $m_1, \ldots, m_N$ and the receiver R wishes to retrieve the message m_c. OT protocols must satisfy the following two security properties: firstly, the sender S does not find out the receiver's choice $c \in \{1, \ldots, N\}$ and, secondly, the receiver only obtains m_c and does not gain any information about the other messages m_i for $i \neq c$. For adaptive k-out-of-N oblivious transfer, $\mathrm{OT}^N_{k\times 1}$, the receiver requests k of these N messages in rounds where the i-th choice is based on the previously obtained messages. We refer the reader to [CNS07, NP05] for more information. Following [CNS07] closely we define *adaptive oblivious transfer* more formally. An adaptive k-out-of-N oblivious transfer scheme $\mathrm{OT}^N_{k\times 1}$ is a tuple of efficient algorithms $(\mathsf{S}_\mathsf{I}, \mathsf{R}_\mathsf{I}, \mathsf{S}_\mathsf{T}, \mathsf{R}_\mathsf{T})$ that consists of an initialization phase and a transfer phase. During the initialization phase the sender and the receiver perform an interactive protocol. In this protocol the sender executes the algorithm S_I on input $(m_1, m_2, \ldots, m_N)$ and the receiver runs the R_I algorithm without input. At the end of the initialization protocol both parties output some (local) state information, denoted by S_0 and R_0, respectively.

Once the initialization phase is over, both parties engage in a transfer protocol. During the i-th transfer, where $1 \leq i \leq k$, the sender runs the algorithm $\mathsf{S}_\mathsf{T}(S_{i-1})$ to obtain some state information S_i whereas the receiver runs the $\mathsf{R}_\mathsf{T}(R_{i-1}, c_i)$ algorithm on input state information R_{i-1} and its choice c_i indicating which message it wishes to receive. The receiver obtains some state information R_i together with the retrieved message m'_{c_i}. A scheme is complete if $m'_{c_i} = m_{c_i}$

for all messages $m_1, \ldots, m_N$, for all selections $c_1, \ldots, c_k \in \{1, \ldots, N\}$ and for all coin tosses of the algorithms. Roughly speaking, security of oblivious transfer demands that the receiver only learns the chosen messages (sender security) and the sender does not know which messages has been chosen (receiver security). In the following we briefly recall (partly verbatim) the security definitions by Camenisch et al. [CNS07]. In contrast to the definition of Naor and Pinkas [NP05] it employs the real-world/ideal-world paradigm for both sender and receiver security (simulatable oblivious transfer). This paradigm compares the execution of an OT protocol in the real-world with an *ideal implementation* (see for example [Can00]). In the real-world experiment, both parties jointly execute the interactive protocol, whereas in the ideal-world the functionality is realized through a trusted third party. Informally, security requires that the malicious receiver/sender gains in the real-world no more information than in the ideal-world. To capture failures one allows the ideal model sender to transmit a bit b, indicating whether the transfer should succeed or abort. We note that this bit is independent of the choice of the receiver, reflecting the fact that the abort should not depend on the receiver's input.

Real Experiment. We begin with the description of the real-world experiment that involves arbitrary sender and receiver algorithms $\mathsf{S}^*_{\mathsf{real}}$ and $\mathsf{R}^*_{\mathsf{real}}$. The experiment $\mathsf{Real}_{\mathsf{S}^*_{\mathsf{real}},\mathsf{R}^*_{\mathsf{real}}}(N, k, m_1, \ldots, m_N, c_1, \ldots, c_k)$ works as follows. Algorithm $\mathsf{S}^*_{\mathsf{real}}$ on input $(m_1, \ldots, m_N)$ interacts with $\mathsf{R}^*_{\mathsf{real}}$ without input. Both parties generate an initial state S_0 and R_0, respectively. Afterwards, the sender and the user perform k interactions. During the i-th execution, for $1 \leq i \leq k$, the sender and receiver interact by running $S_i \leftarrow \mathsf{S}^*_{\mathsf{real}}(S_{i-1})$ and $(R_i, m'_{c_i}) \leftarrow \mathsf{R}^*_{\mathsf{real}}(R_{i-1}, c_i)$ where $c_i \in \{1, \ldots, N\}$ is a message index. It is understood that both algorithm update their state information to S_i and R_i, respectively. Observe that m'_{c_i} and m_{c_i} are not necessarily identical if either party cheats. The output of the experiment $\mathsf{Real}_{\mathsf{S}^*_{\mathsf{real}},\mathsf{R}^*_{\mathsf{real}}}$ is the tuple (S_k, R_k) of the final state information.

Next, we define the behavior of the honest sender and honest user algorithm. That is, the honest sender algorithm $\mathsf{S}_{\mathsf{real}}$ in an $\mathrm{OT}^N_{k\times 1}$ scheme $(\mathsf{S}_{\mathsf{I}}, \mathsf{S}_{\mathsf{T}}, \mathsf{R}_{\mathsf{I}}, \mathsf{R}_{\mathsf{T}})$ takes as input a set of messages $(m_1, \ldots, m_N)$, it runs the algorithm $\mathsf{S}_{\mathsf{I}}(m_1, \ldots, m_N)$ in the initialization phase and runs the S_{T} algorithm in all following interactions. This algorithm always returns $S_k = \epsilon$ as its final state. The honest receiver algorithm $\mathsf{R}_{\mathsf{real}}$ runs the algorithm R_{I} during the initialization phase, the algorithm R_{T} during the transfer phase and stops, outputting the list of received messages $R_k = (m'_{c_1}, \ldots, m'_{c_k})$ as its final state.

Ideal Experiment. In the ideal-world experiment $\mathsf{Ideal}_{\mathsf{S}^*_{\mathsf{ideal}},\mathsf{R}^*_{\mathsf{ideal}}}(N, k, m_1, \ldots, m_N, c_1, \ldots, c_k)$, the (possibly malicious) sender algorithm $\mathsf{S}^*_{\mathsf{ideal}}(m_1, \ldots, m_N)$ generates N messages $(m'_1, m'_2, \ldots, m'_N)$ and hands these over to the trusted party T. In each of the k transfers, T receives a bit b_i and afterwards an index c'_i of the (possibly cheating) receiver $\mathsf{R}^*_{\mathsf{ideal}}$. If $b_i = 1$ and $c'_i \in \{1, \ldots, N\}$, then T sends the message m'_{c_i} to the receiver, and otherwise, $\perp$. The output of the experiment Ideal is the tuple (S_i, R_i) of the final state information of $\mathsf{S}^*_{\mathsf{ideal}}$ and $\mathsf{R}^*_{\mathsf{ideal}}$, respectively.

As above, we now define the honest ideal sender as well as the honest ideal receiver. The honest ideal sender $\mathsf{S}_{\mathsf{ideal}}(m_1, \ldots, m_N)$ sends the messages $m_1, \ldots, m_N$ to the trusted party T in the initialization phase, during the i-th transfer it hands the bit $b_i = 1$ over to T and outputs $S_k = \epsilon$ as its final state. The honest ideal-world receiver $\mathsf{R}_{\mathsf{ideal}}$ submits its real choice $(c_1, c_2, \ldots, c_k)$ to the trusted party and outputs the obtained messages $R_k = (m'_1, m'_2, \ldots, m'_k)$ as its final state.

Sender's security. We say that an $\mathrm{OT}^N_{k\times 1}$ is *sender secure* if for any efficient cheating real-world receiver $\mathsf{R}^*_{\mathsf{real}}$ there exists an efficient ideal-world receiver $\mathsf{R}^*_{\mathsf{ideal}}$ such that for any poly-

nomial $N_q(n)$, any $N \in \{1,\ldots,N_q(n)\}$, any message $m_1, m_2, \ldots, m_N$, and for any choice $c_1, c_2, \ldots, c_k \in \{1,\ldots,N\}$ with $k \in \{1,\ldots,N\}$, the advantage for any efficient distinguisher D in distinguishing the distributions

$$\mathsf{Real}_{\mathsf{S}_{\mathsf{real}},\mathsf{R}^*_{\mathsf{real}}}(N,k,m_1,\ldots,m_N,c_1,\ldots,c_k) \quad \text{and} \quad \mathsf{Ideal}_{\mathsf{S}_{\mathsf{ideal}},\mathsf{R}^*_{\mathsf{ideal}}}(N,k,m_1,\ldots,m_N,c_1,\ldots,c_k)$$

is negligible in n.

Receiver's security. We say that an $\mathrm{OT}^N_{k\times 1}$ is *receiver secure* if for any efficient real-world malicious sender $\mathsf{S}^*_{\mathsf{real}}$, there exists an efficient ideal-world sender $\mathsf{S}^*_{\mathsf{ideal}}$ such that for any polynomial $N_m(q)$, any $N \in \{1,\ldots,N_m(q)\}$, any message $m_1, m_2, \ldots, m_N$, for any choice $c_1, c_2, \ldots, c_k \in \{1,\ldots,N\}$ with $k \in \{1,\ldots,N\}$, the advantage for any efficient distinguisher D in distinguishing the distributions

$$\mathsf{Real}_{\mathsf{S}_{\mathsf{real}},\mathsf{R}^*_{\mathsf{real}}}(N,k,m_1,\ldots,m_N,c_1,\ldots,c_k) \quad \text{and} \quad \mathsf{Ideal}_{\mathsf{S}_{\mathsf{ideal}},\mathsf{R}^*_{\mathsf{ideal}}}(N,k,m_1,\ldots,m_N,c_1,\ldots,c_k)$$

is negligible in n.

6.2 Construction

Our construction, depicted in Figure 2, is a modification of the $\mathrm{OT}^N_{k\times 1}$ protocol of Camenisch et al. and consists of a black-box construction using *any* unique (not necessarily selective-failure) blind signature scheme. The sender in the first step of the protocol generates a key-pair for the blind signature scheme and sends it to the receiver. The receiver, in return, hands N distinct commitments (for values $1, 2, \ldots, N$, represented as n-bit-strings each) over to the sender. These commitments serve as "messages" for the signature generation. Note that distinctiveness of the commitments holds with high probability by the binding property.

After the sender has verified that all commitments are distinct, it encrypts each message in its database by XOR-ing the message m_i with $H(i, s_i)$, where i is the index of the i-th commitment com_i and s_i is the unique signature of message com_i under pk_{BS}. The sender can easily compute this signature locally by running the signature issue protocol with the help of the signing key and an honest user instance for "message" com_i.

After having finished the initialization phase, both parties engage in a transfer phase that consists of a run of the unique blind signature scheme. In the case that the receiver wishes to obtain the i-th message m_i, then it has to choose the commitment com_i (as the message to be signed) during the signature issue protocol.

From a high-level point of view unforgeability guarantees that the receiver cannot receive more messages than interactions took place (sender's security) and blindness guarantees that the sender cannot tell which message has been signed (receiver's security).

6.3 Security of Our Construction

The security of the oblivious transfer protocol follows from the security of the unique blind signature scheme and from the security of the commitment scheme.

Theorem 6.1 *If the unique blind signature scheme* BS *is unforgeable then the* $OT^N_{k\times 1}$ *scheme depicted in Figure 2 is sender-secure in the random oracle model.*

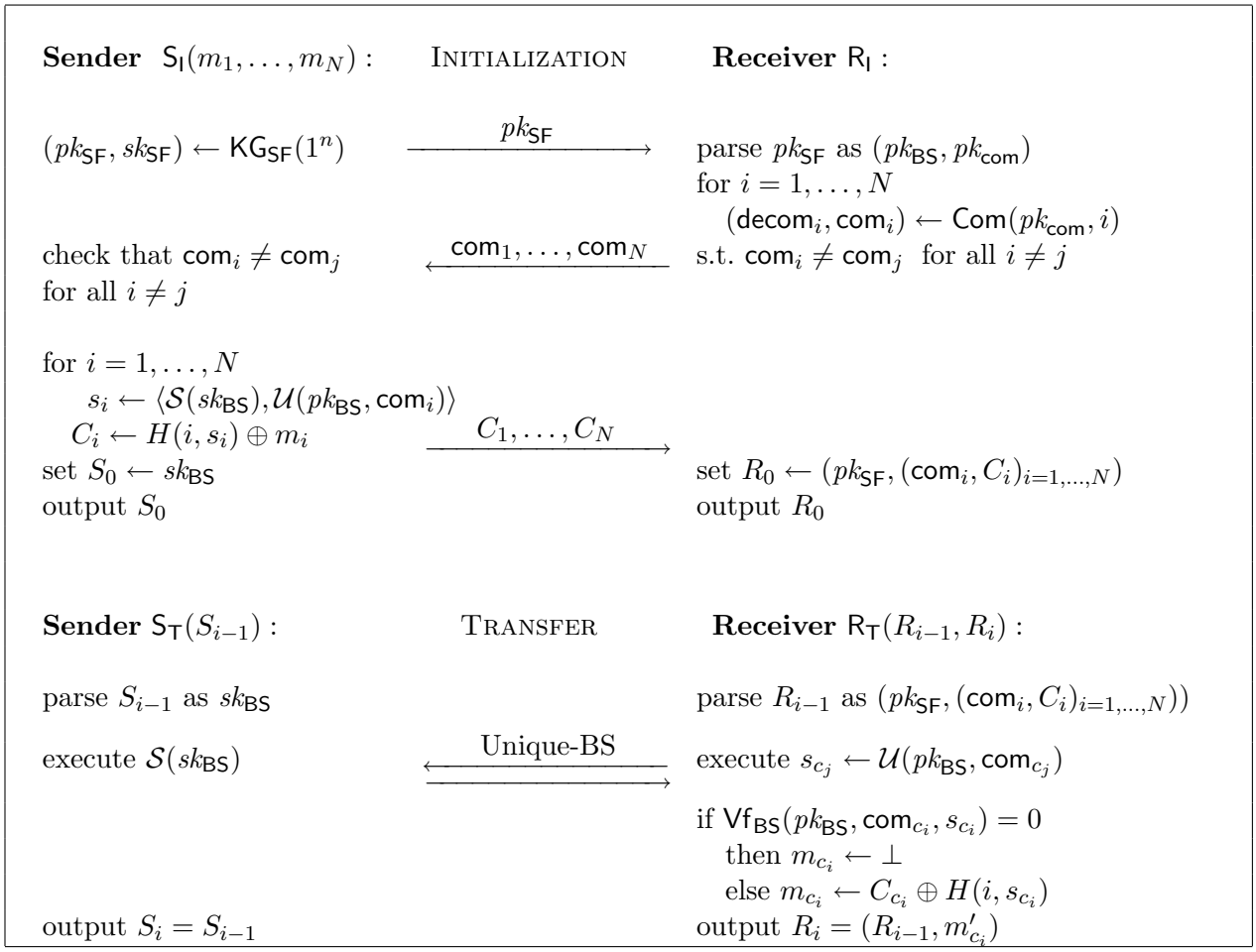

Figure 2: A k-out-of-N oblivious transfer protocol using a random oracle H and any unique blind signature scheme BS.

Proof. In the following, we build for any malicious real-world receiver $\mathsf{R}^*_\mathsf{real}$ an ideal-world receiver $\mathsf{R}^*_\mathsf{ideal}$ that works as follows. The algorithm $\mathsf{R}^*_\mathsf{ideal}$ first generates a key-pair $(sk_\mathsf{SF}, pk_\mathsf{SF}) \leftarrow \mathsf{KG}_\mathsf{SF}(1^n)$ for *any* unique blind signature scheme according to Construction 4.3. It forwards pk_SF to $\mathsf{R}^*_\mathsf{real}$ and receives in return N commitments com_i. Afterwards, $\mathsf{R}^*_\mathsf{real}$ checks wether all commitments are distinct, and if so, it picks N random strings $C_i \leftarrow \{0,1\}^n$ and sends these values to $\mathsf{R}^*_\mathsf{real}$ to obtain the initial state R_0.

During the transfer phase simulates the ideal-world sender $\mathsf{R}^*_\mathsf{ideal}$ the honest signer algorithm of the unique blind signature scheme and engages in k executions of the signature issue protocol with $\mathsf{R}^*_\mathsf{real}$. In order to answer the random oracle queries, the algorithm $\mathsf{R}^*_\mathsf{ideal}$ stores an initially empty associative array $\mathrm{HT}[\cdot]$ together with a counter *ctr*. Whenever $\mathsf{R}^*_\mathsf{real}$ invokes its random oracle $H(\cdot)$ on a value x, then the algorithm $\mathsf{R}^*_\mathsf{ideal}$ returns $\mathrm{HT}[x]$. In the case that this entry in undefined, then $\mathsf{R}^*_\mathsf{ideal}$ proceeds as follows:

> If $x = (i, s_i)$ and $\mathsf{Vf}_\mathsf{BS}(pk_\mathsf{BS}, \mathsf{com}_i, s_i) = 1$ and $i \in [1, N]$ then
> $ctr \leftarrow ctr + 1$; If $ctr > k$, then abort
> Obtain m_i from the ideal functionality

$\mathrm{HT}[x] \leftarrow m_i \oplus C_i$
else $\mathrm{HT}[x] \leftarrow \{0,1\}^\ell$.

Finally, at the end of the simulation, the algorithm $\mathsf{R}^*_{\mathsf{real}}$ outputs its final state which $\mathsf{R}^*_{\mathsf{ideal}}$ also outputs and stops.

It follows easily form the construction of $\mathsf{R}^*_{\mathsf{ideal}}$ that the algorithm is efficient because $\mathsf{R}^*_{\mathsf{real}}$ runs in polynomial time and because the overhead of the key generation, of the simulation of the honest signer algorithm, as well as the overhead of computing the verification equation, can all be performed efficiently. It is also clear that $\mathsf{R}^*_{\mathsf{ideal}}$ performs a perfect simulation of the real-world experiment as long as $\mathsf{R}^*_{\mathsf{ideal}}$ does not abort. Thus, there does *not* exists an distinguisher that is able to distinguish the real-world experiment from the ideal-world experiment with noticeable probability.

In the following we show that $\mathsf{R}^*_{\mathsf{ideal}}$ does not cause $\mathsf{R}^*_{\mathsf{real}}$ to abort with noticeable probability. To do so, let assume towards contradiction that the algorithm $\mathsf{R}^*_{\mathsf{ideal}}$ causes $\mathsf{R}^*_{\mathsf{real}}$ to abort. But then we are able to build a forger F that breaks the unforgeability of BS with noticeable probability. Algorithm F performs a similar simulation of the environment but with two differences. Firstly, the algorithm F does not generate the keys for the blind signature scheme, but forwards the messages between its (external) signing oracle and $\mathsf{R}^*_{\mathsf{real}}$. Secondly, it does not abort if $ctr > k$, but it outputs all $k+1$ tuple (com_i, s_i) where (i, s_i) is the query sent by $\mathsf{R}^*_{\mathsf{real}}$ to its random oracle. Observe that according to our protocol, all commitments have to be distinct and that $\mathsf{R}^*_{\mathsf{real}}$ can engage in at most k transfer protocols. Thus, F outputs $k+1$ different messages together with $k+1$ valid signature after at most k executions of the signature issue protocol. This, however, contradicts the assumption that the blind signature scheme BS is unforgeable. □

Theorem 6.2 *If* BS *is a secure blind signature scheme and* $\mathcal{C}$ *is a secure, length-invariant commitment scheme, then the* $OT^N_{k\times 1}$ *scheme depicted in Figure 2 is receiver-secure in the random oracle model.*

Proof. The proofs follows the one in [CNS07] closely. We have to show that for any efficient cheating real-world sender $\mathsf{S}^*_{\mathsf{real}}$, there exists an efficient ideal-world sender $\mathsf{S}^*_{\mathsf{ideal}}$ such that the outputs of both algorithm are (computational) indistinguishable. This ideal-world algorithm $\mathsf{S}^*_{\mathsf{ideal}}$ works as follows. On input a set of messages $(m_1, m_2, \ldots, m_N)$, algorithm $\mathsf{S}^*_{\mathsf{ideal}}$ executes a black-box simulation of $\mathsf{S}^*_{\mathsf{real}}$ on these messages and answers each random-oracle query by returning random values (but consistently). Let $pk_{\mathsf{SF}} = (pk_{\mathsf{BS}}, pk_{\mathsf{com}})$ be the first outgoing message produced by $\mathsf{S}^*_{\mathsf{real}}$. The attacker $\mathsf{S}^*_{\mathsf{ideal}}$ now computes N distinct commitments $\mathsf{com}_i \leftarrow \mathsf{Com}(pk_{\mathsf{com}}, 0^n)$ and feeds them into $\mathsf{S}^*_{\mathsf{real}}$. Now, consider all random-oracle queries (i, s_i) with $1 \le i \le N$ made by $\mathsf{S}^*_{\mathsf{real}}$. For each query (i, s_i), algorithm $\mathsf{S}'_{\mathsf{real}}$ checks whether $\mathsf{Vf}_{\mathsf{BS}}(pk_{\mathsf{BS}}, \mathsf{com}_j, s_j) = 1$ for some $1 \le j \le N$ and if so, it stores $q_j \leftarrow H(j, s_j)$. During the last move of the initialization phase the algorithm $\mathsf{S}^*_{\mathsf{real}}$ outputs the values $C_1, \ldots, C_N$. Next, algorithm $\mathsf{S}^*_{\mathsf{ideal}}$ sets $m'_i \leftarrow C_i \oplus q_i$ for all $1 \le i \le N$ if q_i is defined, or if q_i is not defined, it sets $m'_i \leftarrow \{0,1\}^n$ to a random value and submits $(m'_1, \ldots, m'_N)$ to the trusted party.

In the following we have to handle the k transfer steps. To handle these queries, $\mathsf{S}^*_{\mathsf{ideal}}$ sets $R_0 \leftarrow (pk_{\mathsf{SF}}, \mathsf{com}_i, C_i)$ for $i = 1, \ldots, N$ and simulates the environment of $\mathsf{S}^*_{\mathsf{real}}$ during the i-th transfer by running $R_i \leftarrow \mathsf{R}_{\mathsf{T}}(R_{i-1}, 1)$, i.e., by always executing the honest receiver algorithm that wishes to receive the message m_1. Observe that $\mathsf{S}^*_{\mathsf{ideal}}$ does not get the choices $(c_1, c_2, \ldots, c_k)$ as input, thus it cannot run R_{T} on the real choice c_i. At the end of the i-th transfer phase, algorithm

$\mathsf{R_T}$ may output $\perp$. In this case, algorithm $\mathsf{S^*_{ideal}}$ submits $b_i = 0$ to the trusted party indicating that this execution should be aborted, and otherwise, it sends $b_i = 1$.

Algorithm $\mathsf{S^*_{ideal}}$ simulates the transfer phase perfectly, i.e., queries of the form (i, s_i) with $1 \leq i \leq N$ and $\mathsf{Vf_{BS}}(pk_{\mathsf{BS}}, i, s) = 1$ are answered with $C_i \oplus m'_i$; other queries are answered with random values (but consistently). Finally, after having terminated k transfer queries, $\mathsf{S^*_{real}}$ outputs some state S_k which $\mathsf{S^*_{ideal}}$ also outputs and stops.

We use a standard hybrid argument (analogously to [CNS07]) to analyze the advantage of an distinguisher D in distinguishing between the experiments $\mathsf{Real}_{\mathsf{S^*_{real}},\mathsf{R_{real}}}$ and $\mathsf{Ideal}_{\mathsf{S^*_{ideal}},\mathsf{R_{ideal}}}$. By $\mathsf{S^*_{ideal,i}}$ we denote an algorithm that simulates the environment of $\mathsf{S^*_{real}}$ identical to $\mathsf{S^*_{ideal}}$ except that it uses $R_i \leftarrow \mathsf{R_T}(R_{i-1}, 1)$ for the first i transfer executions, and which uses $R_i \leftarrow \mathsf{R_T}(R_{i-1}, c_i)$ for the remaining $k - i$ transfers. We further denote by $\mathsf{Game\text{-}i}$ the output of the corresponding experiment, i.e., it contains the final state of $\mathsf{S^*_{ideal,i}}$ as well as the state of the honest ideal receiver $\mathsf{R_{ideal}}(c_1, \ldots, c_k)$ after interacting with the trusted party T. Obviously, $\mathsf{Game\text{-}0}$ corresponds to the real world experiment, i.e., $\mathsf{Game\text{-}0} = \mathsf{Real}_{\mathsf{S^*_{real}},\mathsf{R_{real}}}$ and $\mathsf{Game\text{-}k}$ equals to our ideal-world experiment, i.e., $\mathsf{Game\text{-}k} = \mathsf{Ideal}_{\mathsf{S^*_{ideal}},\mathsf{R_{ideal}}}$. The hybrid argument says that if there exists an efficient algorithm D that is able to distinguish the distributions $\mathsf{Real}_{\mathsf{S^*_{real}},\mathsf{R_{real}}}$ and $\mathsf{Ideal}_{\mathsf{S^*_{ideal}},\mathsf{R_{ideal}}}$ with noticeable advantage ϵ, then there must exist an index $0 \leq i \leq k$ such that D distinguishes $\mathsf{Game\text{-}i}$ and $\mathsf{Game\text{-}(i+1)}$ with probability at least ϵ/k.

Next observe that the proof of Theorem 4.4, which says that every blind signature scheme is selective-failure blind when executed with an a-priori commitment, still holds. To see this note that we distinguish between two cases in this proof. Firstly, if no execution aborts then we can break blindness of the underlying blind signature scheme and, secondly, if at least one execution aborts then it is possible to break secrecy of the commitment scheme (see the proof of Claim 2). In the proof of Claim 2 the adversary receives the commitments of the messages (in random order) at the outset.

In the last step of the proof we have to show that we can use an algorithm D, that distinguishes between the games $\mathsf{Game\text{-}i}$ and $\mathsf{Game\text{-}(i+1)}$, to break selective-failure blindness of BS. To do so, we construct an algorithm $\mathcal{A}_{\mathsf{SF}}$ that runs a black-box simulation of $\mathsf{S^*_{real}}$, that extracts the message and answers all random oracle queries as described for $\mathsf{S^*_{ideal}}$. The algorithm $\mathcal{A}_{\mathsf{SF}}$ simulates the first j-th queries running algorithm $\mathsf{R_T}(\cdot, 1)$, setting $m'_j = \perp$ if the transfer fails, and otherwise $m_j = m_{c_j}$.

During the $(i+1)$-th execution, the algorithm $\mathcal{A}_{\mathsf{SF}}$ behaves as follows. It outputs the tuple $(pk_{\mathsf{BS}}, m_0 = \mathsf{Com}(pk_{\mathsf{com}}, c_i), m_1 = \mathsf{Com}(pk_{\mathsf{com}}, 1))$ according to the first step of the (selective-failure) blindness experiment. In the next step of the experiment, $\mathcal{A}_{\mathsf{SF}}$ interact with two honest user instances $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$ and $\mathcal{U}(pk_{\mathsf{BS}}, m_{1-b})$ for a randomly chosen bit b in the following way. It relays the entire communication of between $\mathsf{S^*_{real}}$ and the first user oracle $\mathcal{U}(pk_{\mathsf{BS}}, m_b)$, whereas it simply aborts the execution with the second oracle.

Algorithm $\mathcal{A}_{\mathsf{SF}}$ eventually receives the signatures (s_0, s_1), encoding the answer right or both. In the case that the issuing in the first execution succeeds, i.e., if $s_0 \neq \perp$, it sets $m'_{i+1} = m_{c_{i+1}}$. Otherwise let $m'_{i+1} = \perp$. We remark that here selective-failure blindness (as opposed to regular blindness) is necessary in order to obtain the information about the left execution.

Analogously to our description above, $\mathcal{A}_{\mathsf{SF}}$ answers the remaining $i + 2 \leq j \leq k$ transfers with the algorithm $\mathsf{R_T}(\cdot, c_j)$, setting $m'_j = m_{c_j}$ if the transfer succeeds, and otherwise, $m'_j = \perp$. Finally, when algorithm $\mathsf{S^*_{real}}$ outputs its final state S_k, then $\mathcal{A}_{\mathsf{SF}}$ runs the distinguisher D on input $(S_k, (m'_1, \ldots, m'_k))$. Note that if $b = 0$, then this tuple is distributed according to $\mathsf{Game\text{-}i}$ and in the case that $b = 1$ it is distributed like $\mathsf{Game\text{-}(i+1)}$. Thus, algorithm $\mathcal{A}_{\mathsf{SF}}$ returns the output of

D and wins the game with probability at least $1/2 + \epsilon/k$. □

Acknowledgments

We thank Heike Busch, Jonathan Katz, and the anonymous reviewers for valuable comments. Both authors are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

References

[Abe01] Masayuki Abe. *A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures*. Advances in Cryptology — Eurocrypt'01, Volume 2045 of Lecture Notes in Computer Science, pages 136–151. Springer-Verlag, 2001.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. *Optimistic Fair Exchange of Digital Signatures*. Advances in Cryptology — Eurocrypt'98, Volume 1403 of Lecture Notes in Computer Science, pages 591–606. Springer-Verlag, 1998.

[Bol03] Alexandra Boldyreva. *Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme*. Public-Key Cryptography (PKC)'03, Volume 2567 of Lecture Notes in Computer Science, pages 31–46. Springer-Verlag, 2003.

[Can00] Ran Canetti. *Security and Composition of Multiparty Cryptographic Protocols*. *Journal of Cryptology*, 13:143–202, 2000.

[Cha83] David Chaum. *Blind Signatures for Untraceable Payments*. Advances in Cryptology — Crypto'82, pages 199–203. Plemum, New York, 1983.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. *Efficient Blind Signatures Without Random Oracles*. Security in Communication Networks, Volume 3352 of Lecture Notes in Computer Science, pages 134–148. Springer-Verlag, 2004.

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. *Simulatable Adaptive Oblivious Transfer*. Advances in Cryptology — Eurocrypt'07, Lecture Notes in Computer Science, pages 573–590. Springer-Verlag, 2007.

[DPP97] Ivan Damgård, Torben Pedersen, and Birgit Pfitzmann. *On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures*. Volume 10, pages 163–194. Springer-Verlag, 1997.

[Fis06] Marc Fischlin. *Round-Optimal Composable Blind Signatures in the Common Reference String Model*. Advances in Cryptology — Crypto'06, Volume 4117 of Lecture Notes in Computer Science, pages 60–77. Springer-Verlag, 2006.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. *A Practical Secret Voting Scheme for Large Scale Elections*. ASIACRYPT '92: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Lecture Notes in Computer Science, pages 244–251, London, UK, 1993. Springer-Verlag.

[GMPY06] Juan Garay, Philip MacKenzie, Manoj Prabhakaran, and Ke Yang. *Resource Fairness and Composability of Cryptographic Protocols.* Theory of Cryptography Conference (TCC)'06, Volume 3876 of Lecture Notes in Computer Science, pages 404–428. Springer-Verlag, 2006.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. *Invariant Signatures and Non-Interactive Zero-Knowledge Proofs are Equivalent.* CRYPTO, Lecture Notes in Computer Science, pages 228–245. Springer-Verlag, 1992.

[Gol04] Oded Goldreich. *The Foundations of Cryptography*, Volume 2. Cambridge University Press, 2004.

[HK07] Omer Horvitz and Jonathan Katz. *Universally-Composable Two-Party Computation in Two Rounds.* Advances in Cryptology — Crypto'07, Lecture Notes in Computer Science, pages 111–129. Springer-Verlag, 2007.

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. *Concurrently-Secure Blind Signatures Without Random Oracles or Setup Assumptions.* Theory of Cryptography Conference (TCC)'07, Volume 4392 of Lecture Notes in Computer Science, pages 323–341. Springer-Verlag, 2007.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. *Security of Blind Digital Signatures.* Advances in Cryptology — Crypto'97, Volume 1294 of Lecture Notes in Computer Science, pages 150–164. Springer-Verlag, 1997.

[KZ05] Aggelos Kiayias and Hong-Sheng Zhou. *Two-Round Concurrent Blind Signatures without Random Oracles.* Number 2005/435 in Cryptology eprint archive. `eprint.iacr.org`, 2005.

[KZ08] Aggelos Kiayias and Hong-Sheng Zhou. *Equivocal Blind Signatures and Adaptive UC-Security.* Theory of Cryptography Conference (TCC)'08, Lecture Notes in Computer Science. Springer-Verlag, 2008.

[Nao91] Moni Naor. *Bit Commitment Using Pseudo-Randomness. Journal of Cryptology*, 4(2):151–158, 1991.

[NP05] Moni Naor and Benny Pinkas. *Computationally Secure Oblivious Transfer. Journal of Cryptology*, 18(1):1–35, 2005.

[Oka06] Tatsuaki Okamoto. *Efficient Blind and Partially Blind Signatures Without Random Oracles.* Theory of Cryptography Conference (TCC)'06, Volume 3876 of Lecture Notes in Computer Science, pages 80–99. Springer-Verlag, 2006.

[PS00] David Pointcheval and Jacques Stern. *Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology*, 13(3):361–396, 2000.

[Rab81] Michael Rabin. *How to Exchange Secrets by Oblivious Transfer.* Technical Report TR-81, Aiken Computation Laboratory, 1981.

A Alternative Definition for Multi-Execution Selective-Failure Blindness

As mentioned before, an alternative way to define multi-execution selective-failure blindness is to let the adversary associate a particular execution to a message-signature pair. We refer to this definition as multi-execution selective-failure blindness (according to prediction) or, for short, prediction security. Analogously, we call a multi-execution selective-failure blind (according to ordering) signature scheme as an ordering-secure blind signature scheme as defined in Section 3.3:

Definition A.1 *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}_{\mathsf{BS}}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf}_{\mathsf{BS}})$ *is called* multi-execution selective-failure blind (according to prediction) *if it is unforgeable (as in Definition 2.2) and the following holds:*

Prediction Security. *For any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find, issue, *and* reveal*) the probability that experiment* $\mathsf{PMSFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$ *returns* 1 *is negligibly close to* $\frac{1}{q-r}$*, where* q *is the number of protocol executions and* r *the number of reveal queries. The experiment is defined as follows.*

Experiment $\mathsf{PMSFBlind}^{\mathsf{BS}}_{\mathcal{S}^*}(n)$

$(pk_{\mathsf{BS}}, M, \beta_{\mathit{find}}) \leftarrow \mathcal{S}^*(\mathit{find}, 1^n)$ *where* $M = (m_1, \ldots, m_q)$ *with* $m_i \in \{0,1\}^n$

Select a random permutation π *over* $\{1, 2, \ldots, q\}$

$\beta_{\mathit{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}) \rangle^1, \ldots, \langle \cdot, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)}) \rangle^1, \mathsf{Reveal}(\cdot, \pi, st^{\mathit{rev}})}(\mathit{issue}, \beta_{\mathit{find}})$

and let $\sigma_{\pi(1)}, \ldots, \sigma_{\pi(q)}$ *denote the (possibly undefined) local outputs of* $\mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(1)}), \ldots, \mathcal{U}(pk_{\mathsf{BS}}, m_{\pi(q)})$*, immediately stored in* st^{rev} *once an execution finishes (*st^{rev} *is initially set to* $(\perp, \ldots, \perp)$*);* $\mathsf{Reveal}(\cdot, \pi, st^{\mathit{rev}})$ *is an oracle which on input* i *returns* $(\pi(i), st^{\mathit{rev}}_i)$.

Return to $\mathcal{S}^*$ *all signatures* $v = (\sigma_1, \ldots, \sigma_q)$ *iff all executions have yielded valid signatures; otherwise return a vector* $v \in \{0,1\}^q$ *where the* i*-th entry is* 1 *if the* i*-th signature is valid, and* 0 *otherwise.*

$(i, j) \leftarrow \mathcal{S}^{*, \mathsf{Reveal}(\cdot, \pi, st^{\mathit{rev}})}(\mathit{reveal}, v, \beta_{\mathit{issue}})$

Return 1 *iff* $\pi(i) = j$*, and* $\mathcal{S}^*$ *has never queried* Reveal *about* i*, and* $r \leq q - 2$.

We next show that this definition is equivalent to multi-execution selective-failure blindness (according to ordering). For one direction we give a direct proof and the other direction follows easily from the equivalence to the two-execution case for ordering-based security:

Proposition A.2 *An ordering-secure blind signature scheme is also prediction-secure.*

Proof. The proof idea is as follows. Suppose that there exists an adversary breaking the prediction property of the scheme BS. This adversary reveals some executions and outputs a pair (i, j) predicting $\pi(i) = j$. We then build an algorithm which determines the order of two executions i, i' by exploiting the prediction $\pi(i) = j$ and comparing j with the remaining image $\pi(i') = j'$.

Description of Adversary $\mathcal{S}^*$. More precisely, let $\mathcal{A}$ be an adversary breaking multi-execution selective-failure blindness (according to prediction) of BS with a probability noticeable greater than $\frac{1}{q-r}$. By q we denote the number of protocol executions and by r the number of reveal oracle queries

made by $\mathcal{A}$. We construct an algorithm $\mathcal{S}^*$ against the ordering property, which runs a black-box simulation of $\mathcal{A}$ and works as follows.

In the first step of this simulation the algorithm $\mathcal{A}$ outputs a vector M of q messages together with a public key pk_{BS}. The attacker $\mathcal{S}^*$ outputs this pair and has now access to q external user instances as well as to a reveal oracle. The i-th user instance is initialized with the pair $(pk_{\mathsf{BS}}, m_{\pi(i)})$. It relays the entire communication between its external user instances, between its reveal oracle, and $\mathcal{A}$. During these executions, $\mathcal{A}$ may try to invoke the reveal oracle more than $q-2$ times. If this happens, $\mathcal{S}^*$ stops before passing on the $(q-1)$-st query and outputs the two indices corresponding to the two unrevealed executions in random order.

At the end of the q executions, algorithm $\mathcal{S}^*$ is given a vector $(m_1, \sigma_1), \ldots, (m_q, \sigma_q)$ of q message-signature pairs, or, if at least one execution aborted, a vector $v \in \{0,1\}^q$. The i-th entry of the vector v indicates if the i-th protocol instance yielded a valid signature. It forwards the given vector to $\mathcal{A}$ which, at the end of the simulation, returns two values (i, j) trying to predict $\pi(i) = j$.

In case that $r < q-2$ then $\mathcal{S}^*$ queries its reveal oracle $q-2-r$ times by picking each unqueried index with the same probability (but not i). If one of the $q-2-r$ revealed values is equal to j, then $\mathcal{S}^*$ stops and returns the two unrevealed indices (i, i') in a randomly chosen order. Otherwise, the algorithm $\mathcal{S}^*$ outputs (i, i') iff $j < j'$ and (i', i) else, where $j' \neq j$ denotes the last unrevealed image under π which is not equal to the prediction j.

Analysis of Adversary $\mathcal{S}^*$. First note that the probability that $\mathcal{A}$ returns two values (i, j) such that $\pi(i) = j$ is

$$\delta(n) \geq \frac{1}{q-r} + \epsilon(n)$$

where $0 \leq r \leq q-2$ denotes the number of oracle queries made be $\mathcal{A}$ and $\epsilon(n)$ is noticeable. The success probability $\rho(n)$ of $\mathcal{S}^*$ is composed of two different events:

$$\rho(n) := \mathrm{Prob}[\mathsf{Succ}(\mathcal{S}^*) \wedge \mathsf{Succ}(\mathcal{A})] + \mathrm{Prob}[\mathsf{Succ}(\mathcal{S}^*) \wedge \neg\mathsf{Succ}(\mathcal{A})]\,.$$

The first event $\mathsf{Succ}(\mathcal{S}^*) \wedge \mathsf{Succ}(\mathcal{A})$ is the case where $\mathcal{S}^*$ and $\mathcal{A}$ are both successful. As the algorithm $\mathcal{S}^*$ performs a perfect simulation from $\mathcal{A}$'s point of view, the overall probability of this event is $\delta(n)$.

The second event $\mathsf{Succ}(\mathcal{S}^*) \wedge \neg\mathsf{Succ}(\mathcal{A})$, where only $\mathcal{S}^*$ is successful, consists of two parts. First, we have the case that $\mathcal{A}$ tries to invoke the reveal oracle more than $q-2$ times. Second, there is the case that $\mathcal{S}^*$ finds an index which is equal to j, which means that $\mathcal{S}^*$ discovers that $\pi(i) \neq j$. Note that, given $\mathcal{A}$ does not win and $r \leq q-2$, the probability that $\mathcal{S}^*$ picks an index which returns j is $1 - \frac{1}{q-r-1}$. As the algorithm $\mathcal{S}^*$ in both cases returns the two unrevealed indices in a random order, the overall success probability of $\mathcal{S}^*$ is bounded from below by $(1-\delta(n)) \cdot (1 - \frac{1}{q-r-1}) \cdot \frac{1}{2}$.

Putting things together we get

$$\begin{aligned}
\rho(n) &= \mathrm{Prob}[\mathsf{Succ}(\mathcal{S}^*) \wedge \mathsf{Succ}(\mathcal{A})] + \mathrm{Prob}[\mathsf{Succ}(\mathcal{S}^*) \wedge \neg\mathsf{Succ}(\mathcal{A})] \\
&\geq \delta(n) \cdot 1 + (1 - \delta(n)) \cdot \left(1 - \frac{1}{q-r-1}\right) \cdot \frac{1}{2} \\
&\geq \frac{1}{2} + \frac{\delta(n)}{2} - \frac{1}{2(q-r-1)} + \frac{1}{2(q-r-1)(q-r)} \\
&\geq \frac{1}{2} + \frac{\epsilon(n)}{2} + \frac{1}{2(q-r)} - \frac{1}{2(q-r-1)} + \frac{1}{2(q-r-1)(q-r)} \\
&\geq \frac{1}{2} + \frac{\epsilon(n)}{2} \qquad ,
\end{aligned}$$

which is noticeably greater than $\frac{1}{2}$. □

The following corollary shows the remaining direction for the equivalence of the two notions:

Corollary A.3 *A prediction-secure blind signature scheme is also ordering-secure.*

Proof. It is easy to see that a prediction-secure blind signature scheme (for multiple executions) is also selective-failure blind (for two executions). Then, a selective-failure blind signature scheme (for two executions) is also ordering-secure (for many executions) by Proposition 3.5. □