Concurrently-Secure Blind Signatures without Random Oracles or Setup Assumptions*

Carmit Hazay[1], Jonathan Katz[2], Chiu-Yuen Koo[2], and Yehuda Lindell[1]

[1] Bar-Ilan University.
{harelc,lindell}@cs.biu.ac.il
[2] University of Maryland.
{jkatz,cykoo}@cs.umd.edu

Abstract. We show a new protocol for blind signatures in which security is preserved even under arbitrarily-many concurrent executions. The protocol can be based on standard cryptographic assumptions and is the first to be proven secure in a concurrent setting (under *any* assumptions) without random oracles or a trusted setup assumption such as a common reference string. Along the way, we also introduce new definitions of security for blind signature schemes.

1 Introduction

Blind signature schemes, introduced by Chaum [11], are a fascinating primitive that (roughly speaking) enable a user to interact with a signer and obtain a signature on a message m without revealing anything about m to the signer. Blind signature schemes are a crucial component of many systems in which certain values need to be *certified*, yet *anonymity* should be ensured: classical examples include e-cash (where a bank signs 'e-coins' that are withdrawn by customers) and e-voting (where an authority signs public keys for voters to use when they later cast their votes).

Definitions of security for blind signature schemes were first proposed by Pointcheval and Stern [29], though many refinements and extensions of their original definitions have since been suggested. At a high level, all existing definitions impose two basic requirements: *blindness* (or anonymity) and *unforgeability*. Blindness formalizes the notion that a malicious signer should be unable to 'link' any message/signature pair with a particular execution of the signing protocol. Unforgeability for blind signatures is the analogue of the notion of unforgeability for standard signature schemes: informally, a malicious user should be unable to output a valid signature on any message other than those whose signatures were explicitly requested from the signer. A subtlety in the case of blind signatures is that a malicious user's execution of the protocol with the

* This research was supported by US-Israel Binational Science Foundation grant #2004240. Work of the first author was also supported by an Eshkol fellowship from the Israel Ministry of Science and Technology. Work of the second author was also supported by NSF CAREER award #0447075.

signer may not result in any well-defined message whose signature is being requested. Because of this, the formal definition requires that for any polynomial ℓ and any user executing the protocol ℓ times with the signer, the user should be unable to output $\ell + 1$ valid signatures on $\ell + 1$ distinct messages.

When defining blindness and unforgeability it is necessary to distinguish whether security requires different executions of the protocol to be carried out *sequentially* (i.e., waiting for one execution to finish before beginning the next), or whether security holds even when multiple executions are performed *concurrently* (i.e., in an arbitrarily-interleaved manner). (One can also consider the intermediate case in which executions are run *in parallel.*) Concurrency in the context of blindness has received little attention, both because the 'standard' definition of blindness considers only two executions of the protocol and also, perhaps, because many known constructions of blind signature schemes achieve perfect blindness. In contrast, handling concurrency in the context of unforgeability has received much attention (surveyed below), and it is not hard to see that — assuming there exist blind signature schemes at all — there exist schemes that are unforgeable in the sequential setting but *not* in a concurrent setting.

1.1 Previous Constructions

Chaum [11] proposed a candidate blind signature scheme without any proof of security (though his scheme was later proven secure in the random oracle model under a somewhat non-standard cryptographic assumption [5]). Since then, numerous works have aimed to design secure schemes. We review these here, with particular attention to the type of unforgeability proved.

Schemes in the random oracle model. Initial constructions of blind signature schemes were in the random oracle model [6], and, in fact, until relatively recently all efficient constructions relied on random oracles. Pointcheval and Stern [28] showed the first secure blind signature schemes, though they prove unforgeability (in the parallel setting) only for a user who requests *logarithmically*-many signatures. This was improved in later work by Pointcheval [27], who showed schemes that are unforgeable (in a restricted variant of the parallel setting) for polynomially-many signatures. Abe [1] gave a protocol with improved round complexity, and also proved unforgeability in the concurrent setting. Bellare, et al. [5] and Boldyreva [8] present 2-round blind signature schemes; note that 2-round protocols (which consist of a single message from the user and a response by the signer) are automatically secure in a concurrent setting.

Schemes in the standard model. Relatively early, it was suggested [12] that blind signatures might be constructed using protocols for generic secure 2-party computation. Juels, Luby, and Ostrovsky [19] point out that the naïve way of implementing this approach does not work, but show how to adapt and extend this idea so as to achieve a secure solution. Although they claim security in the concurrent setting, no details of the proof in this case are provided; as best as we can tell, their solution is secure in the sequential setting only. Indeed, security of their protocol in the concurrent setting seems to require a *concurrently-secure* protocol for 2-party computation, but constructing such protocols without random oracles or setup assumptions is currently a major open question. The work of [4] could be used here, but then security would require sub-exponential hardness assumptions (something avoided in our work).

Camenisch, et al. [9] show the first *efficient* protocol secure in the standard model, proven unforgeable only for the case of sequential attacks.

Lindell [23] has shown the impossibility of concurrently-secure blind signatures if simulation-based definitions of security are used.[1] In an effort to overcome the limitations of the above protocols, as well as Lindell's impossibility result, much recent work has focused on proving security for blind signature schemes in the concurrent setting by assuming a *common reference string* [26, 21, 16]. However, although Lindell's impossibility result was used as justification for relying on a common reference string in these works, Lindell's results do not apply if *game-based* security definitions (rather than *simulation-based* security definitions) are used. Indeed, this serves as the starting point for our work.[2]

1.2 Our Contributions

As hinted at earlier, the standard definition of blindness considers only the interaction of a malicious signer with two users; furthermore, the definition does not seem to reasonably extend for the case of multiple users (the issue is how to deal with a signer who may abort some sessions). We propose a new definition here which extends seamlessly to the multi-user setting, and (in retrospect) seems to capture better the security requirements of a blind signature scheme.

As our main contribution, we present the first concurrently-secure blind signature scheme that does not rely on random oracles or any setup assumptions such as a common reference string. In order to 'bypass' the impossibility result of Lindell [23], we prove security using game-based definitions that have anyway been standard in almost all prior work in this area. Our protocol relies on standard cryptographic assumptions (e.g., trapdoor permutations and the decisional Diffie-Hellman assumption), and we prove security with respect to game-based definitions that are stronger than others that have appeared in the literature.

Besides being interesting in its own right, our construction serves as yet another illustration that known impossibility results for concurrently-secure 2-party computation [23, 24] might be overcome for *specific* functionalities of interest by considering relaxed (yet still meaningful) definitions of security. In this sense, our work exemplifies what we see as a viable alternative to the approaches to concurrently-secure computation taken by, e.g., [10, 23, 31, 3, 20, 4, 25], who focus on staying within the simulation paradigm (in part, because they are striving for a generic result) but are thus forced to impose additional assumptions (e.g., a common reference string [10] or a bound on network delay [20]) or to settle for alternate definitional relaxations (e.g., bounded concurrency [23] or super-polynomial-time simulation [4]).

[1] Technically, he only rules out *black-box* proofs of security.

[2] We do not formally define what it means for a definition to be 'simulation-based' or 'game-based,' but instead appeal to the reader's intuition regarding such matters.

1.3 Outline

In Section 2 we discuss definitions of security for blind signature schemes, and present a new set of definitions that are stronger than any to have previously appeared in the literature. We also propose, for the first time, a definition of blindness for the case of a signer interacting with an arbitrary number of users.

We then build up to our main result in stages: in Section 3.1 we describe the recent blind signature scheme of Fischlin [16] which is used as a building block in our work, and then in Section 3.2 we construct a blind signature scheme that can be proven concurrently-secure using *complexity leveraging*.[3] Our main result (which does not rely on complexity leveraging) appears in Section 4, along with proof sketches of the blindness and unforgeability properties. Due to space limitations, complete proofs are omitted but will appear in the full version.

2 Definitions

A standard signature scheme is a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, where the *key generation algorithm* Gen takes as input a security parameter 1^k and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$ with the security parameter implicit in both; the *signing algorithm* Sign takes as input a message m and a secret key sk and outputs a signature σ; and the *verification algorithm* Vrfy takes as input a public key pk, a message m, and a candidate signature σ and outputs a decision bit. Correctness requires that if $(\mathsf{pk}, \mathsf{sk})$ is output by $\mathsf{Gen}(1^k)$ then $\mathsf{Vrfy}_{\mathsf{pk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1$ for all m. We use the standard definition of existential unforgeability under adaptive chosen-message attacks [18].

We assume signature schemes that are *length-regular*: i.e., there exists a polynomial $p(\cdot)$ such that if $(\mathsf{pk}, \mathsf{sk})$ are output by $\mathsf{Gen}(1^k)$ then for any m in the message space (1) $\mathsf{Sign}_{\mathsf{sk}}(m) \in \{0,1\}^{p(|m|)}$ and (2) $\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma) = 0$ if $\sigma \notin \{0,1\}^{p(|m|)}$. We will not write this explicitly in the rest of the paper.

We now define a *blind* signature scheme.

Definition 1. *A* blind signature scheme *consists of* PPT *algorithms* $\mathsf{Gen}, \mathsf{Vrfy}$ *along with interactive* PPT *algorithms* $\mathcal{S}, \mathcal{U}$ *such that:*

- Gen*, on input* 1^k*, outputs a key pair* $(\mathsf{PK}, \mathsf{SK})$ *with* k *implicit in both.*
- *The joint execution of* $\mathcal{S}$*, holding input* SK*, and* $\mathcal{U}$*, holding inputs* PK, m*, results in an output* σ *for* $\mathcal{U}$*, assuming neither* $\mathcal{S}$ *nor* $\mathcal{U}$ *abort. We write this as* $\sigma \leftarrow \langle \mathcal{S}_{\mathsf{SK}}, \mathcal{U}_{\mathsf{PK}}(m) \rangle$*. If* $\mathcal{U}$ *aborts, its output is* $\perp$ *(which is never a valid signature) and we assume that it notifies* $\mathcal{S}$*.*
- Vrfy*, on input* PK, m, σ*, outputs a decision bit.*

Correctness requires that for all $(\mathsf{PK}, \mathsf{SK})$ *output by* $\mathsf{Gen}(1^k)$ *and all* m*, if* $\sigma \leftarrow \langle \mathcal{S}_{\mathsf{SK}}, \mathcal{U}_{\mathsf{PK}}(m) \rangle$ *then* $\mathsf{Vrfy}_{\mathsf{PK}}(m, \sigma) = 1$*.*

[3] Roughly speaking, this means we assume primitives A and B such that A cannot be broken in polynomial time but can be broken in time $T(k)$ for some super-polynomial function T, while B cannot be broken in time $T(k)$.

We now define *unforgeability* and *blindness*. In both definitions, the adversary maintains state throughout its execution.

Definition 2. *Blind signature scheme* $(\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *is* unforgeable *if for any polynomial* ℓ*, the success probability of any* PPT *algorithm* $\hat{\mathcal{U}}$ *in the following game is negligible:*

- $\mathsf{Gen}(1^k)$ *outputs keys* $(\mathsf{PK}, \mathsf{SK})$*, and* $\hat{\mathcal{U}}$ *is given* PK*.*
- $\hat{\mathcal{U}}(\mathsf{PK})$ *interacts concurrently with* $\ell = \ell(k)$ *instances* $\mathcal{S}^1_{\mathsf{SK}}, \ldots, S^\ell_{\mathsf{SK}}$*.*
- $\hat{\mathcal{U}}$ *outputs* $(m_1, \sigma_1, \ldots, m_{\ell+1}, \sigma_{\ell+1})$*.*

$\hat{\mathcal{U}}$ succeeds *if the* $\{m_i\}$ *are distinct and* $\mathsf{Vrfy}_{\mathsf{PK}}(m_i, \sigma_i) = 1$ *for all* i*.*

We next turn to defining blindness. We begin with a (strong) variant of the standard definition of blindness, which only considers the execution of the signer with two users. This is followed by some discussion of how the definition might be extended for the case of multiple users.

Definition 3. *Blind signature scheme* $(\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *satisfies* blindness *if the advantage of any* PPT *algorithm* $\hat{\mathcal{S}}$ *in the following game is negligible:*

1. $\hat{\mathcal{S}}(1^k)$ *outputs an arbitrary public key* PK *along with equal-length messages* m_0, m_1*.*
2. *A random bit* b *is chosen, and* $\hat{\mathcal{S}}$ *interacts concurrently with* $\mathcal{U}_b \stackrel{\text{def}}{=} \mathcal{U}_{\mathsf{PK}}(m_b)$ *and* $\mathcal{U}_{\bar{b}} \stackrel{\text{def}}{=} \mathcal{U}_{\mathsf{PK}}(m_{\bar{b}})$*. When* $\mathcal{U}_b, \mathcal{U}_{\bar{b}}$ *have completed their execution,* σ_0, σ_1 *are defined as follows:*
   - *If* either $\mathcal{U}_b$ or $\mathcal{U}_{\bar{b}}$ *abort, then* $(\sigma_0, \sigma_1) := (\perp, \perp)$*.*
   - *Otherwise, let* σ_0 *(resp,* σ_1*) be the output of* $\mathcal{U}_0$ *(resp.,* $\mathcal{U}_1$*).*

   $\hat{\mathcal{S}}$ *is given* (σ_0, σ_1)*.*
3. *Finally,* $\hat{\mathcal{S}}$ *outputs a bit* b'*.*

$\hat{\mathcal{S}}$ succeeds *(denoted Succ) if* $b' = b$*. The* advantage *of* $\hat{\mathcal{S}}$ *is* $\left|\Pr[\mathsf{Succ}] - \frac{1}{2}\right|$*.*

For the definition to be meaningful, we cannot give $\hat{\mathcal{S}}$ the signature output by one user in case the other aborts: if we did, $\hat{\mathcal{S}}$ could simply abort the execution with its 'left' oracle and then, depending on whether it is given a signature on m_0 or m_1, easily determine b. On the other hand, in contrast to [16], we allow the game to continue if either user aborts (this only strengthens the definition). Note also that $\hat{\mathcal{S}}$ may generate PK in an arbitrary manner, not necessarily using Gen. It seems perfectly natural to us to allow this possibility, though it appears to have been formally considered only relatively recently [2, 26, 16].

In extending the above definition to the case of a signer interacting with an arbitrary number of users, an obvious approach is to allow the signer to output two *vectors* $\boldsymbol{m}_0, \boldsymbol{m}_1$ containing the same messages $m_1, \ldots, m_\ell$ (possibly allowing repeats) in permuted order. A difficulty that arises is how to deal with a signer who aborts some of the sessions. Some natural ways of dealing with this are (1) if the signer aborts any session, it receives no signatures; or (2) say $\boldsymbol{m}_0 = (m^0_1, \ldots, m^0_\ell)$ and $\boldsymbol{m}_1 = (m^1_1, \ldots, m^1_\ell)$. Then if the signer aborts the i^{th}

session, it is given neither the signature on m_i^0 nor the signature on m_i^1. The first option seems (to us) to be too weak. The second option seems a bit arbitrary, though reasonable; an aesthetic drawback is that it is not clear that it is implied by Definition 3. In the full version we sketch a third possibility, intermediate in strength between the above two, which *is* implied by Definition 3.

In any case, all the above ways of dealing with abort (even in the original case with two users) seem a bit arbitrary even though for technical reasons they are necessary to make the definitions non-trivial. We therefore propose a new definition which, in our opinion, handles the issue of abort in a cleaner way. Though it allows some 'attacks' which are ruled out by Definition 3, we believe it models the security desired of typical proposed applications of blind signatures (such as e-cash or e-voting). Further discussion follows the definition.

Definition 4. *Blind signature scheme* $(\mathsf{Gen}, \mathcal{S}, \mathcal{U}, \mathsf{Vrfy})$ *satisfies* a posteriori blindness *if for any polynomial ℓ, any ℓ' such that $1 \leq \ell'(k) \leq \ell(k)$ for all k, and any* PPT *algorithm $\hat{\mathcal{S}}$, the advantage of $\hat{\mathcal{S}}$ in the following game is at most a negligible quantity:*

1. *$\hat{\mathcal{S}}(1^k)$ outputs an arbitrary public key PK and a message distribution*[4] *$\mathcal{M}$ sampleable in polynomial time.*
2. *Messages $m_1, \ldots, m_\ell$ are sampled according to $\mathcal{M}$, and $\hat{\mathcal{S}}$ interacts concurrently with $\mathcal{U}_{\mathsf{PK}}(m_1)$, …, $\mathcal{U}_{\mathsf{PK}}(m_\ell)$. The game ends if the number of non-aborted sessions is not equal to ℓ'. Otherwise, we say event $\mathsf{NA}(\ell')$ occurs and the game continues.*
3. *Let $i_1, \ldots, i_{\ell'}$ denote the indices of the non-aborted sessions and let π be a random one-to-one function mapping $\{1, \ldots, \ell'\}$ to these indices. $\hat{\mathcal{S}}$ is given $(m_{\pi(1)}, \sigma_{\pi(1)})$, …, $(m_{\pi(\ell')}, \sigma_{\pi(\ell')})$.*
4. *Finally, $\hat{\mathcal{S}}$ outputs (i, i').*

$\hat{\mathcal{S}}$ succeeds *(this event is denoted by* Succ*) if $\pi(i) = i'$. The* advantage *of $\hat{\mathcal{S}}$ is* $\Pr[\mathsf{Succ}] - \frac{1}{\ell'} \Pr[\mathsf{NA}(\ell')]$.

Note that allowing the signer to choose the message distribution is stronger than quantifying over all sampleable distributions, since it allows the signer to choose a distribution that depends on the (maliciously-chosen) public key.

The intent of the above definition is to model the scenario where (honest) users anyway choose the 'messages' to be signed from some known distribution. For example, in the case of e-cash the message might be a random string; in the case of e-voting the message might be an honestly-generated public key; finally, a scenario similar (but not identical) to that of Definition 3 can be achieved if $\mathcal{M}$ is the uniform distribution over $\{m_0, m_1\}$. After interacting with users who choose their messages according to this distribution, the signer is given all message/signature pairs (in a randomly-permuted order) from the non-aborted sessions; this corresponds to the scenario when the users in the non-aborted sessions reveal their message/signature pairs (e.g., by spending an e-coin or casting

[4] This could be specified, e.g., by a circuit whose output (on uniform input) defines the distribution.

a vote). Informally, the signer 'wins' if it can link some message/signature pair to its corresponding session with probability better than randomly guessing a non-aborted session.

The nice thing about the above definition is that it models exactly what the signer actually 'sees' in the real world, without imposing any artificial (though necessary) restrictions. We remark also that Definition 4 in the special case $\ell = 2$ implies the general case.

We stress, however, that Definition 4 guarantees no 'blindness' whatsoever in the aborted sessions. In particular, a scheme in which the user reveals m (and aborts) if the signer sends an improper first message could still potentially be secure with respect to Definition 4 though it would not satisfy Definition 3. We do not view this as a problem since we view 'messages' as having no inherent secrecy requirement (indeed, the user eventually reveals its message anyway); rather, the goal is to prevent the linking of a particular message (that is later used) to a particular session. In this sense, schemes satisfying a posteriori blindness are analogous to commitment schemes with a posteriori secrecy (cf. [17, Section 4.8.2.5]). For this reason, schemes satisfying this notion may not be appropriate for all possible applications of blind signatures.[5]

3 A Warm-Up for Our Main Result

Our blind signature scheme builds on an elegant construction due to Fischlin [16] that relies on a common reference string. We review Fischlin's scheme and then, as a step toward our main result, present a blind signature scheme that can be proven concurrently-secure using complexity leveraging (cf. footnote 3).

3.1 Fischlin's Blind Signature Scheme

We describe a simplified[6] version of Fischlin's scheme that satisfies our definitions of blindness and unforgeability in the *common reference string* (CRS) model. Let $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ be a standard signature scheme, and let Com be a perfectly-binding commitment scheme. Fischlin's scheme is defined as follows (see also Figure 1):

Setup: The CRS contains a public key pk_E for a semantically-secure public-key encryption scheme, and a string ρ used as a CRS for a non-interactive zero-knowledge (NIZK) proof system. $\mathcal{E}_{pk_E}(\cdot)$ denotes encryption using pk_E.

Key generation: $\mathsf{Gen}(1^k)$ runs $\mathsf{Gen}'(1^k)$ to obtain keys $(\mathsf{pk}', \mathsf{sk}')$ and outputs these keys.

Signing: The protocol for a user $\mathcal{U}$ to obtain a signature on a message m is as follows:

[5] However, we conjecture that any scheme satisfying Def. 4 can be converted to one satisfying Def. 3 by using a commitment to the message in the signing protocol.

[6] The scheme presented by Fischlin includes some additional complications that are used to achieve *strong* unforgeability, which we do not consider here.

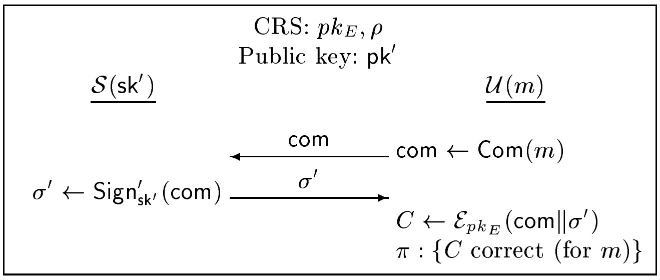

Fig. 1. Fischlin's protocol.

- $\mathcal{U}$ computes $\mathsf{com} \leftarrow \mathsf{Com}(m)$ and sends com to the signer.
- $\mathcal{S}$ computes $\sigma' \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}(\mathsf{com})$ and sends σ' to $\mathcal{U}$.
- $\mathcal{U}$ verifies the signature sent in the previous step, and aborts if it is invalid. Otherwise, the user computes $C \leftarrow \mathcal{E}_{pk_E}(\mathsf{com}\|\sigma')$ and computes an NIZK proof π (using ρ) that $(m, C, pk_E, \mathsf{pk}') \in L$ where L is defined as the set of tuples $(m, C, pk_E, \mathsf{pk}')$ for which there exists $\omega_1, \omega_2, \mathsf{com}, \sigma'$ such that

$$\mathsf{com} := \mathsf{Com}(m; \omega_1) \bigwedge C := \mathcal{E}_{pk_E}(\mathsf{com}\|\sigma'; \omega_2) \bigwedge \mathsf{Vrfy}'_{\mathsf{pk}'}(\mathsf{com}, \sigma') = 1 .$$

(Note that L is an $\mathcal{NP}$ language.) The signature is (C, π).

Verification: To verify signature (C, π) on m with respect to public key pk' and CRS (pk_E, ρ), verify that π is a valid proof (with respect to ρ) that $(m, C, pk_E, \mathsf{pk}') \in L$.

We now sketch the proofs of blindness and unforgeability. For blindness, note that the signer observes only a commitment to m, an encryption of this commitment, and an NIZK proof π; it is not too hard to see that none of these leaks information about m, nor allows the signer to correlate a particular execution of the protocol with a particular signature output by $\mathcal{U}$.

For unforgeability, an adversary $\hat{\mathcal{U}}$ that forges a signature in the sense of Definition 2 can be used to construct a forger $\mathcal{F}$ for standard signature scheme Π': given public key pk' of an instance of Π', forger $\mathcal{F}$ generates pk_E on its own (along with the corresponding secret key sk_E), generates ρ at random, and runs $\hat{\mathcal{U}}$ in the natural way. $\mathcal{F}$ can easily execute the protocol with $\hat{\mathcal{U}}$ using its own signing oracle. Finally, if $\hat{\mathcal{U}}$ outputs $\ell + 1$ distinct messages $\{m_i\}$ with valid signatures $\{(C_i, \pi_i)\}$, then with all but negligible probability (by soundness of the NIZK proof system and perfect binding of the commitment scheme) each C_i is a valid encryption of a *distinct* commitment com_i and a valid signature σ'_i (with respect to Π') on this commitment. Given this, $\mathcal{F}$ can recover all the $\{(\mathsf{com}_i, \sigma'_i)\}$ by decrypting all the ciphertexts using sk_E; since $\mathcal{F}$ accessed its signing oracle exactly ℓ times, at least one $(\mathsf{com}_i, \sigma'_i)$ leads to a valid forgery for Π'.

3.2 Concurrently-Secure Blind Signatures: a Partial Solution

If we try to adapt Fischlin's scheme so as to avoid the CRS, we encounter two main obstacles. We describe these now, along with our solutions.

Removing ρ: If the signer generates ρ, the proof π may leak information about the underlying m, com, or σ' (which would violate blindness); on the other hand, the user clearly cannot generate ρ itself since then soundness may no longer apply and forgery would be possible.

We can resolve this by relying on ZAPs [14] rather than NIZK, and having the signer include the first message ρ for a ZAP as part of its public key. (A ZAP is a two-round witness-indistinguishable proof system; see Appendix A). Since a ZAP is witness indistinguishable but not zero knowledge, however, the protocol must be changed so as to provide an alternate witness that will be available to a simulator (for proving blindness) but not to a malicious user (or else forgery becomes possible). We provide such a witness by having the signer include $y_0 = f(x_0)$ and $y_1 = f(x_1)$ in its public key, where f is a one-way function, and then having the signer give a witness-indistinguishable proof of knowledge of either x_0 or x_1 as part of the signing protocol [15]. When constructing the signature (after execution of the signing protocol), the user $\mathcal{U}$ computes C as in Fischlin's protocol and then gives a witness-indistinguishable proof π that (essentially) it either constructed C appropriately or it knows one of x_0 or x_1.

Removing pk_E: If the signer generates pk_E then it is trivial for a malicious signer to violate the blindness property; if the user generates pk_E on its own, then the reduction in the proof of unforgeability given in the previous section no longer works since $\mathcal{F}$ can no longer recover a forgery for Π' from a forgery for the blind signature scheme (since it cannot decrypt C).

If we are willing to rely on complexity leveraging, we can overcome this difficulty by using a *commitment scheme* Com^* to construct C rather than an *encryption scheme*. If Com^* is secure against PPT adversaries, blindness still holds. If, however, Com^* can be broken in time $T(k)$ for some super-polynomial function $T(\cdot)$, then (referring to the proof of unforgeability in the previous section) we can construct a forger $\mathcal{F}$ running in time $O(T(k))$ who extracts a valid signature for Π'. If we further assume that Π' is secure *even against adversaries running in time* $O(T(k))$, this still yields a contradiction and is enough to prove unforgeability of the blind signature scheme.

This gives the main intuition. We now give a more complete description of the protocol, along with sketches of the proofs of blindness and unforgeability. We take the liberty of being somewhat informal, as this protocol is meant mainly as a 'stepping stone' toward our main result (which does not use complexity leveraging).

Let $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ be a standard signature scheme, and let f be a one-way function. We assume these are secure (in the appropriate sense) for adversaries running in time $O(T(k))$, where $T(\cdot)$ is a super-polynomial function. Let $\mathsf{Com}, \mathsf{Com}^*$ be perfectly-binding commitment schemes, where Com^* is such

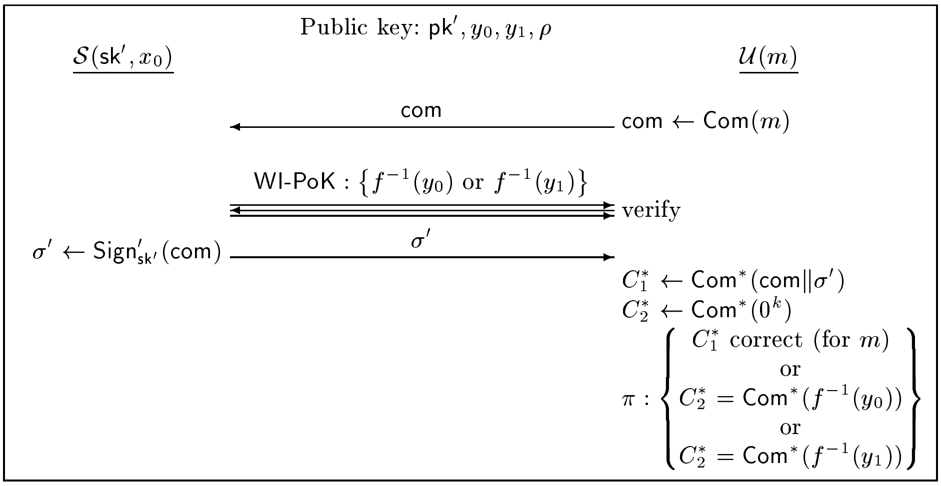

Fig. 2. A partial solution using complexity leveraging.

that given $C^* \leftarrow \mathsf{Com}^*(m)$ it is possible to recover m in time $T(k)$. (However, Com^* is still hiding for PPT adversaries.) Our protocol is defined as follows:

Key generation $\mathsf{Gen}(1^k)$ runs $\mathsf{Gen}'(1^k)$ to obtain keys $(\mathsf{pk}', \mathsf{sk}')$. It also chooses $x_0, x_1 \leftarrow \{0,1\}^k$ and sets $y_0 := f(x_0)$ and $y_1 := f(x_1)$. Finally, it computes ρ as the verifier's initial message in a ZAP. The public key is $\mathsf{PK} := (\mathsf{pk}', y_0, y_1, \rho)$ and the secret key is $\mathsf{SK} := (\mathsf{sk}', x_0)$.

Signing The protocol for $\mathcal{U}$ to obtain a signature on message m is as follows:

- $\mathcal{U}$ computes $\mathsf{com} \leftarrow \mathsf{Com}(m)$ and sends com to the signer.
- $\mathcal{S}$ and $\mathcal{U}$ execute a witness-indistinguishable proof of knowledge (WI-PoK) in which $\mathcal{S}$ proves knowledge of either $f^{-1}(y_0)$ or $f^{-1}(y_1)$. (This should be witness indistinguishable even against adversaries running in $O(T(k))$ time.) If this proof fails, $\mathcal{U}$ aborts.
- $\mathcal{S}$ computes $\sigma' \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}(\mathsf{com})$ and sends σ' to $\mathcal{U}$.
- $\mathcal{U}$ verifies the signature sent in the previous step, and aborts if it is invalid. Otherwise, the user computes $C_1^* \leftarrow \mathsf{Com}^*(\mathsf{com}\|\sigma')$ and $C_2^* \leftarrow \mathsf{Com}^*(0^k)$. It then computes a ZAP π (with respect to ρ) that $(m, C_1^*, C_2^*, \mathsf{pk}', y_0, y_1) \in L$, where L contains tuples for which there exist $\omega_1, \omega_2, \mathsf{com}, x, \sigma'$ such that:

$$\mathsf{com} := \mathsf{Com}(m; \omega_1) \bigwedge C_1^* := \mathsf{Com}^*(\mathsf{com}\|\sigma'; \omega_2) \bigwedge \mathsf{Vrfy}'_{\mathsf{pk}'}(\mathsf{com}, \sigma') = 1$$
$$or$$
$$C_2^* := \mathsf{Com}^*(x; \omega_2) \bigwedge f(x) \in \{y_0, y_1\}$$

(Note that $L \in \mathcal{NP}$.) The signature is (C_1^*, C_2^*, π).

Verification To verify signature (C_1^*, C_2^*, π) on message m, verify that π is a valid proof (with respect to ρ) that $(m, C_1^*, C_2^*, \mathsf{pk}', y_0, y_1) \in L$.

We now sketch the proofs of blindness and unforgeability. Again, these are informal because they are mostly intended to provide the reader with some intuition toward our main result that appears in the following section.

Proof sketch (blindness) Given a malicious signer $\hat{\mathcal{S}}$ we will consider a sequence of hybrid experiments, and argue that the success probability of $\hat{\mathcal{S}}$ (in the sense of Definition 3) cannot change by more than a negligible amount in going from one experiment to the next. The first experiment is the original game of Definition 3, and in the final experiment the success probability of $\hat{\mathcal{S}}$ will be exactly $1/2$. We conclude that the success probability of $\hat{\mathcal{S}}$ in the original experiment is negligibly-close to $1/2$, thus proving blindness.

In the initial experiment H_0 the signer $\hat{\mathcal{S}}$ outputs a public key $\mathsf{PK} = (\mathsf{pk}', y_0, y_1, \rho)$ and two equal-length messages m_0, m_1. A random bit b is chosen and $\hat{\mathcal{S}}$ interacts with $\mathcal{U}_b \stackrel{\text{def}}{=} \mathcal{U}_{\mathsf{PK}}(m_b)$ and $\mathcal{U}_{\bar{b}} \stackrel{\text{def}}{=} \mathcal{U}_{\mathsf{PK}}(m_{\bar{b}})$. If neither of these users aborts, then $\hat{\mathcal{S}}$ is given the signatures output by these users. Finally, $\hat{\mathcal{S}}$ outputs a bit b', and succeeds if $b' = b$.

In the first hybrid experiment H_1, whenever $\mathcal{U}_0$ does not abort we extract from the $\mathsf{WI\text{-}PoK}$ (given by $\hat{\mathcal{S}}$ to $\mathcal{U}_0$) a value x such that $f(x) \in \{y_0, y_1\}$. If $\hat{\mathcal{S}}$ gives a valid $\mathsf{WI\text{-}PoK}$ but extraction fails, b' is chosen at random; otherwise, b' is computed as in H_0. Clearly, the success probabilities in games H_0 and H_1 differ by only a negligible amount. We remark that extraction here is only required from *one* of the proofs given by $\hat{\mathcal{S}}$, and furthermore if the $\mathsf{WI\text{-}PoK}$ given to $\mathcal{U}_0$ fails then no signatures need be provided to $\hat{\mathcal{S}}$ (even if the $\mathsf{WI\text{-}PoK}$ given to $\mathcal{U}_1$ succeeds). Thus, no difficulties arise due to the concurrent execution of two $\mathsf{WI\text{-}PoK}$s by $\hat{\mathcal{S}}$.

In H_2, the signatures output by $\mathcal{U}_0, \mathcal{U}_1$ are both computed using the witness x that was extracted (this is only done if neither user aborts and extraction is successful, as otherwise either $\hat{\mathcal{S}}$ is given $(\perp, \perp)$ or else extraction failed and b' is chosen at random). Specifically, each user computes C_1^* as before but now sets $C_2^* := \mathsf{Com}^*(x; \omega)$; the proof π is constructed using (ω, x) as the witness. Hiding of Com^* (for PPT adversaries) and witness-indistinguishability of the ZAP imply that the success probabilities of $\hat{\mathcal{S}}$ in experiments H_1 and H_2 differ by only a negligible amount.

In the final experiment H_3, the first component C_1^* of the signature generated by each user is computed as a commitment to 'garbage', i.e., an all-0s string of the appropriate length. Also, the commitments com sent by each of the users during their execution of the protocol are replaced with commitments to garbage as well. Hiding of Com and Com^* (against PPT adversaries) again implies that the success probabilities in experiments H_2 and H_3 differ by only a negligible amount.

In H_3, both protocol executions are distributed identically and both signatures are independent of these executions; thus, the probability of success is exactly $1/2$. This concludes the proof. ■

Proof sketch (unforgeability) As in the analysis of the Fischlin scheme, an adversary $\hat{\mathcal{U}}$ that, with non-negligible probability, forges a signature with respect to the blind signature scheme can be used as a sub-routine of an algorithm that 'breaks' another cryptographic assumption. Here, however, there are two main differences:

1. First, the resulting algorithm must be able to extract the underlying messages being committed to in C_1^* and/or C_2^*; this can be done in time $T(k)$ (but *not* in polynomial time) and so we obtain an algorithm running in $O(T(k))$ time rather than in polynomial time.
2. Second, the algorithm is only ensured to extract (with non-negligible probability) *either* $\ell + 1$ distinct commitments $\{\mathsf{com}_i\}$ along with $\ell + 1$ valid signatures $\{\sigma'_i\}$, *or* a value x with $f(x) \in \{y_0, y_1\}$ (in the proof for the Fischlin scheme only the first of these could occur). The first event immediately leads to a forgery on Π'. The second event leads to an algorithm $\mathcal{I}$ inverting f with non-negligible probability (using the technique of Feige and Shamir [15]).

If the signature scheme Π' and the one-way function f are secure even against adversaries running in time $O(T(k))$, the above leads to a contradiction. Hence, we conclude that the blind signature scheme is unforgeable. ∎

4 A Concurrently-Secure Blind Signature Scheme

In this section, we describe our main result: a concurrently-secure blind signature scheme based on standard cryptographic assumptions. In addition to a standard signature scheme, our construction also relies on a perfectly-binding commitment scheme and a ZAP, reviewed in Appendix A. We also use a special type of commitment scheme, described below, and a particular concurrent zero-knowledge protocol, discussed in detail in the following section.

For our protocol we will require a special type of commitment scheme that we call *ambiguous*. In such a scheme, commitment depends on a key pk_c which can be generated in one of two ways: either by a 'normal' key-generation procedure ComGen, or by an 'alternate' key-generation procedure ExtGen which outputs some additional trapdoor information td along with pk_c. If pk_c is generated by ComGen, the scheme is perfectly hiding. On the other hand, if pk_c is generated by ExtGen then td enables extraction of the committed value. Formally:

Definition 5. *An* ambiguous commitment scheme *is a tuple of* PPT *algorithms* (ComGen, ExtGen, Com, Extract) *such that:*

Functionality: $\mathsf{ComGen}(1^k)$ *outputs a key* pk_c. $\mathsf{ExtGen}(1^k)$ *outputs a key* pk_c *and a trapdoor* td.

Indistinguishability: *The keys output by* ComGen *and* ExtGen *are computationally indistinguishable; that is:*

$$\left\{pk_c \leftarrow \mathsf{ComGen}(1^k) : pk_c\right\} \overset{c}{\approx} \left\{(pk_c, \mathsf{td}) \leftarrow \mathsf{ExtGen}(1^k) : pk_c\right\}.$$

Perfect hiding: *If* pk_c *is output by* ComGen*, then (with probability 1)* $\mathsf{Com}_{pk_c}(\cdot)$ *is a perfectly-hiding commitment scheme.*

Extraction: *If* (pk_c, td) *is output by* ExtGen*, then* $\mathsf{Extract}_{\mathsf{td}}(\mathsf{Com}_{pk_c}(m)) = m$ *with probability 1. (This implies that if* pk_c *is output by* ExtGen*, then* $\mathsf{Com}_{pk_c}(\cdot)$ *is perfectly binding.)*

The last two requirements imply that the ranges of ComGen and ExtGen are disjoint.

Commitment schemes with the above functionality (satisfying also some additional requirements) were shown previously by Damgård and Nielsen [13] based on a variety of number-theoretic assumptions. The following construction is easily seen to satisfy Definition 5 under the decisional Diffie-Hellman assumption:

– $\mathsf{ComGen}(1^k)$ first generates a group $\mathbb{G}$ of prime order q, along with generators $g, h \in \mathbb{G}$. It then chooses $r_1, r_2 \leftarrow \mathbb{Z}_q$. If $r_1 \neq r_2$ it outputs $pk_c = (\mathbb{G}, q, g, h, g^{r_1}, h^{r_2})$, and otherwise[7] it outputs $pk_c = (\mathbb{G}, q, g, h, g^0, h^1)$.
– $\mathsf{ExtGen}(1^k)$ generates $\mathbb{G}, q, g, h$ exactly as ComGen. It then chooses $r \leftarrow \mathbb{Z}_q$ and outputs $pk_c = (\mathbb{G}, q, g, h, g^r, h^r)$ and $\mathsf{td} = r$.
– $\mathsf{Com}^*_{pk_c}(m)$, where $m \in \mathbb{G}$ and $pk_c = (\mathbb{G}, q, g, h, g_1, h_1)$, chooses random $x, y \leftarrow \mathbb{Z}_q$ and outputs $A = g^x h^y$ and $B = g_1^x h_1^y \cdot m$.
– $\mathsf{Extract}_r(A, B)$ outputs B/A^r.

4.1 The PRS Concurrent Zero-Knowledge Protocol

As part of our blind signature scheme, we rely on a concurrent zero-knowledge protocol adapted from work of Prabhakaran, Rosen, and Sahai [30, 32] and described in Figure 3; we will refer to this protocol as cZK. Protocol cZK is almost identical to the protocol shown in [32, Section 4.8.2], with one difference being that we are satisfied with an *argument* system[8] rather than a *proof* system. The first step of the second stage of cZK is also added specifically for the proof of security of our blind signature scheme. Finally, cZK is also a (stand-alone) *argument of knowledge*, something we need for our protocol.

We do not offer a proof that cZK satisfies the definition of concurrent zero-knowledge, appealing instead to the analysis in [32] which extends without significant modification to our protocol. Actually, for the proof of security of our blind signature scheme we do not rely on the concurrent zero-knowledge property of cZK as a 'black-box,' but instead rely on the properties of the *specific* zero-knowledge simulator shown by Prabhakaran, et al. We therefore briefly describe their simulation strategy at a high level.

The keys to the simulation strategy of [30] are that (1) second-stage messages can be simulated (without knowing a witness) in a straight-line manner as long as the simulator learns in advance the value α that the verifier committed to in the first phase; and (2) the value α can be extracted if the verifier ever answers correctly for two different values of s^j. Correspondingly, the simulation

[7] We explicitly check whether $r_1 \neq r_2$, even though this occurs with negligible probability, since perfect hiding for keys output by ComGen must hold with probability 1.

[8] Recall that in a proof system soundness must hold unconditionally, while in an argument system soundness need only hold against a PPT cheating prover.

Inputs: The prover and verifier reduce their common input to a graph $G = (V, E)$. From its witness, the prover computes (as private input) a Hamiltonian cycle $C \subseteq E$. Let k be the security parameter.

First stage: Let $r = \log^2(k)$.

1. The verifier uniformly selects $\alpha \in \{0,1\}^r$, and then chooses values $\{\alpha^0_{i,j}\}^r_{i,j=1}$ and $\{\alpha^1_{i,j}\}^r_{i,j=1}$ at random subject to the constraint that $\alpha^0_{i,j} \oplus \alpha^1_{i,j} = \alpha$ for all i, j. The verifier sets $\mathsf{com} \leftarrow \mathsf{Com}(\alpha)$ and $\mathsf{com}^b_{i,j} \leftarrow \mathsf{Com}(\alpha^b_{i,j})$, and sends all these commitments to the prover.
2. For $j = 1, \ldots, r$:
   1. The prover selects a random $s^j \in \{0,1\}^r$ and sends it to the verifier.
   2. Let $s^j = s^j_1 \cdots s^j_r$. The verifier sends $\{\alpha^{s^j_i}_{i,j}\}^r_{i=1}$ along with the randomness used in generating $\{\mathsf{com}^{s^j_i}_{i,j}\}^r_{i=1}$. The prover verifies that these match the corresponding initial commitments sent by the verifier, and aborts if this is not the case.

Second stage: The prover and verifier run r parallel executions of (a modified version of) Blum's Hamiltonicity protocol [7]:

1. The verifier and prover execute a (standard) zero-knowledge proof in which the verifier proves that its commitments (sent in step 1 of the first phase) are 'consistent': namely, that there exist values α and $\{\alpha^0_{i,j}, \alpha^1_{i,j}\}^r_{i,j=1}$ such that (1) com is a commitment to α; (2) $\mathsf{com}^b_{i,j}$ is a commitment to $\alpha^b_{i,j}$ for all i, j, b; and (3) $\alpha^0_{i,j} \oplus \alpha^1_{i,j} = \alpha$ for all i, j. If the verifier's proof fails, the prover aborts.
2. The prover selects r random permutations $\pi_1, \ldots, \pi_r$ of the vertices V, and sends perfectly-binding commitments to the entries of the adjacency matrices of the resulting permuted graphs.
3. The verifier sends α, and the verifier and prover execute a (standard) zero-knowledge proof in which the verifier proves that com is a commitment to α. If the verifier's proof fails, the prover aborts.
4. For $j = 1, \ldots, r$ do: if $\alpha_j = 1$ send π_j and open all the commitments in the j^{th} adjacency matrix. If $\alpha_j = 0$ open only the commitments to entries corresponding to the (permuted) cycle C.
5. The verifier checks the values sent by the prover in the standard way.

Fig. 3. A concurrent zero-knowledge argument of knowledge.

used in [30, 32] can be separated, both conceptually and functionally, into two parts: a 'look-ahead' sub-routine (whose goal is to extract α for all existing sessions) and a 'straight-line simulation' sub-routine (which actually generates the transcript that is output by the simulator). The look-ahead sub-routine dynamically updates a table containing (roughly speaking) all the α-values that have been extracted thus far; if the straight-line simulation sub-routine is reached and a corresponding value of α (needed to continue the simulation) is not in the table, the simulator aborts with output $\perp$.

Another important feature of the simulation strategy is that control alternates between the two sub-routines according to a *fixed* schedule that does not depend on the actions of the particular verifier under consideration. This, in turn, means that we can distinguish in advance the portion of the simulator's random coins that are used for 'look-aheads' and those that are used for straight-line simulation. We will exploit this feature in the unforegablility proof of our protocol. We remark also that the transcript generated by the 'straight-line simulation' sub-routine is built up incrementally, message-by-message, but once a message is placed in this transcript it is never removed.

4.2 Our Construction: an Overview

We begin with some intuition motivating our construction. Recalling the scheme presented in Section 3.2, we see that the use of complexity leveraging there is due to the need to extract from the commitments of $\hat{\mathcal{U}}$ in the proof of unforgeability (which requires super-polynomial time). A first thought is to let Com^* in that protocol be an *ambiguous* commitment scheme, with the public key pk_c for the commitment included in the signer's public key and generated using ComGen. Then, in the proof of unforgeability, we can generate pk_c using ExtGen (instead of ComGen) and thus extract the necessary values from the signature forgeries output by $\hat{\mathcal{U}}$.

An immediate problem is that a malicious signer could then easily violate blindness by generating pk_c using ExtGen. To prevent this, we have the signer provide a proof[9] that pk_c was correctly generated as part the signing protocol. Because we will want to replace pk_c with an incorrectly-generated key in the proof of unforgeability, this proof will need to be *(concurrent) zero knowledge* (witness indistinguishability does not help us here). Because we will again want to provide an 'alternate' witness in the proof of blindness, it will also be a proof of knowledge. We remark that once we introduce this change, we no longer need the values y_0, y_1 in the signer's public key

This almost completes the description of our protocol. However, a difficulty arises if we try to prove unforgeability of the construction as described to this point. Roughly speaking, for the construction thus far it is possible to prove the following:

> *Given $\hat{\mathcal{U}}$ who interacts with ℓ instances of $\mathcal{S}$ and outputs $\ell+1$ valid signatures on distinct messages with non-negligible probability (cf. Definition 2), we can construct an adversarial forger $\mathcal{F}$ who interacts with a signing oracle for (standard signature scheme) Π' and outputs $\ell+1$ valid signatures on distinct messages with non-negligible probability.*

The problem is that $\mathcal{F}$ *makes more than ℓ queries to its signing oracle*, and it is therefore not clear that the $\ell+1$ signatures output by $\mathcal{F}$ yield a valid forgery! To see why, note that although $\hat{\mathcal{U}}$ invokes only ℓ instances of $\mathcal{S}$, simulation of the zero-knowledge proof by $\mathcal{F}$ requires rewinding of $\hat{\mathcal{U}}$, and many more than ℓ

[9] Actually, we use an argument system but this does not affect the intuition.

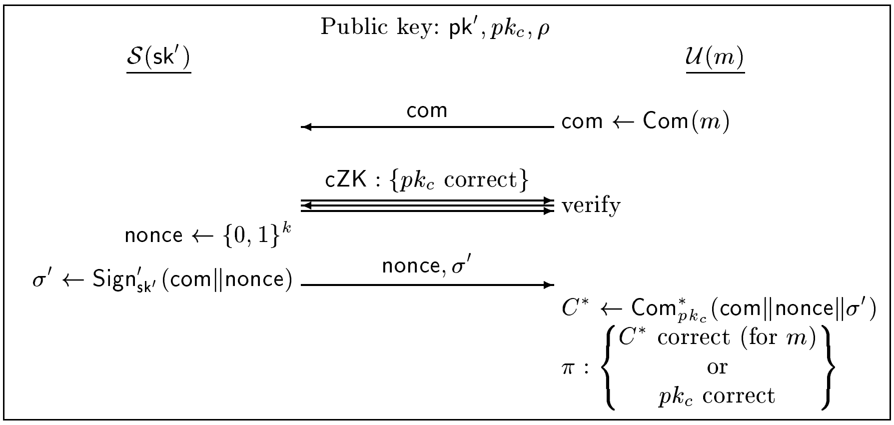

Fig. 4. A high-level overview of our protocol.

signatures will have to be generated as part of this rewinding. (In the protocol of Section 3.2 no rewinding was needed and so $\mathcal{F}$ made exactly ℓ queries to its signing oracle there.) Dealing with this issue is the most difficult and technically-involved aspect of our construction.

We resolve the issue in the following way: instead of having the signer generate a (standard) signature on the commitment com sent by the user in the first round, we have the signer choose a random string $\mathsf{nonce} \in \{0,1\}^k$ and sign $\mathsf{com}\|\mathsf{nonce}$ (computation of the final signature by $\mathcal{U}$ is changed in the obvious way). In the proof of unforgeability, we still construct a forger $\mathcal{F}$ who outputs $\ell + 1$ valid signatures on distinct messages $\{(\mathsf{com}_i\|\mathsf{nonce}_i)\}$, but requests *more* than ℓ signatures from its signing oracle. Now, however, we can show that these $\ell + 1$ messages are (in a certain sense) *independent* of the random nonces used during the rewinding done by $\mathcal{F}$. (Here, in particular, we rely on the fact that in step 1 of the second phase of cZK the verifier proves consistency of its commitments, and therefore it does not matter in which iteration the simulator extracted α.) Since the nonces used during rewinding are chosen at random, this means that with overwhelming probability at least one of the messages $\mathsf{com}_i\|\mathsf{nonce}_i$ will be different from any query made by $\mathcal{F}$ to its signing oracle, in which case $\mathcal{F}$ can output a forgery for Π'.

We remark that in proving the above we rely on the specific concurrent zero-knowledge protocol cZK, as well as a particular simulation strategy for this protocol, rather than relying on concurrent zero-knowledge in a 'black-box' way. Indeed, we do not know how to prove unforgeability of our construction when instantiated with an arbitrary concurrent zero-knowledge protocol.

4.3 Our Construction

We now give the details of our construction. Let $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ be a standard signature scheme, let cZK be the protocol of Figure 3, and let (ComGen,

ExtGen, Com^*, $\mathsf{Extract}$) be an ambiguous commitment scheme. Our protocol is constructed as follows (see Figure 4):

Key generation First, $\mathsf{Gen}'(1^k)$ is run to obtain keys $(\mathsf{pk}', \mathsf{sk}')$ and $\mathsf{ComGen}(1^k)$ is run to obtain pk_c. The signer also computes ρ as the verifier's initial message in a ZAP. The public key is $\mathsf{PK} := (\mathsf{pk}', pk_c, \rho)$ and the secret key is sk' along with the randomness used to generate pk_c.

Signing The protocol for a user $\mathcal{U}$ to obtain a signature on a message m is as follows:

- $\mathcal{U}$ computes $\mathsf{com} \leftarrow \mathsf{Com}(m)$ and sends com to the signer.
- $\mathcal{S}$ and $\mathcal{U}$ execute protocol cZK by which $\mathcal{S}$ proves that pk_c was generated correctly. Formally, it proves that $pk_c \in L_{\mathsf{ComGen}}$, where

$$L_{\mathsf{ComGen}} \stackrel{\text{def}}{=} \left\{ pk_c : \exists \omega \text{ s.t. } pk_c := \mathsf{ComGen}(1^k; \omega) \right\}.$$

  If this proof fails, $\mathcal{U}$ aborts. If $\mathcal{S}$ aborts in cZK (because it detects that $\mathcal{U}$ is cheating), then $\mathcal{S}$ aborts the entire signing protocol.
- $\mathcal{S}$ chooses $\mathsf{nonce} \leftarrow \{0,1\}^k$, computes $\sigma' \leftarrow \mathsf{Sign}'_{\mathsf{sk}'}(\mathsf{com}\|\mathsf{nonce})$, and sends nonce, σ' to $\mathcal{U}$.
- $\mathcal{U}$ verifies the signature sent in the previous step, and aborts if it is invalid. Otherwise, the user computes $C^* \leftarrow \mathsf{Com}^*_{pk_c}(\mathsf{com}\|\mathsf{nonce}\|\sigma')$. It then computes a ZAP π (with respect to ρ) that $(m, C^*, \mathsf{pk}', pk_c) \in L_2$, where L_2 contains tuples such that there exist $\omega_1, \omega_2, \mathsf{com}, \mathsf{nonce}, \sigma'$ with:

$$\begin{gathered} \Big(\mathsf{com} := \mathsf{Com}(m; \omega_1) \bigwedge \\ C^* := \mathsf{Com}^*_{pk_c}(\mathsf{com}\|\mathsf{nonce}\|\sigma'; \omega_2) \bigwedge \mathsf{Vrfy}'_{\mathsf{pk}'}(\mathsf{com}\|\mathsf{nonce}, \sigma') = 1 \Big) \\ \textit{or} \\ pk_c := \mathsf{ComGen}(1^k; \omega_1) \end{gathered}$$

  The signature is (C^*, π).

Verification To verify signature (C^*, π) on message m, verify that π is a valid proof (with respect to initial message ρ) that $(m, C^*, \mathsf{pk}', pk_c) \in L_2$.

We claim the following about the above scheme:

Theorem 1. *Assuming that (1)* Com *is computationally hiding; (2)* (ComGen, ExtGen, Com^*, $\mathsf{Extract}$) *is an ambiguous commitment scheme; (3)* cZK *is an argument of knowledge with negligible knowledge error; and (4) the ZAP being used is witness indistinguishable, the blind signature scheme above satisfies blindness.*

Theorem 2. *Assuming that (1)* Com *is perfectly binding; (2)* (ComGen, ExtGen, Com^*, $\mathsf{Extract}$) *is an ambiguous commitment scheme; (3) the ZAP being used has negligible soundness error; and (4)* $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is existentially unforgeable under adaptive chosen-message attacks, the blind signature scheme above satisfies unforgeability.*

The proof of blindness (in the sense of Definition 3) follows the general structure of the proof of blindness sketched in Section 3.2. (The above scheme also satisfies all definitions of blindness mentioned in Section 2 and in particular Definition 4.) The proof of unforgeability was sketched in Section 4.2. Complete proofs of all the above will appear in the full version.

References

1. M. Abe. A Secure Three-Move Blind Signature Scheme for Polynomially-Many Signatures. Eurocrypt 2001.
2. M. Abdalla, C. Namprempre, and G. Neven. On the (Im)possibility of Blind Message Authentication Codes. CT-RSA 2006.
3. B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. FOCS 2004.
4. B. Barak and A. Sahai. How To Play Almost Any Mental Game Over The Net — Concurrent Composition via Super-Polynomial Simulation. FOCS 2005.
5. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *J. Cryptology* 16(3): 185–215 (2003).
6. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. ACM CCCS '93.
7. M. Blum. How to Prove a Theorem so No One Else Can Claim It. *Proceedings of the International Congress of Mathematicians*, pp. 1444–1451, 1986.
8. A. Boldyreva. Efficient Threshold Signatures, Multisignatures, and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. PKC 2003.
9. J. Camenisch, M. Koprowski, and B. Warinschi. Efficient Blind Signatures without Random Oracles. SCN 2004.
10. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. STOC 2002.
11. D. Chaum. Blind Signatures for Untraceable Payments. Crypto '82.
12. I. Damgård. Payment Systems and Credential Mechanisms with Provable Security against Abuse by Individuals. Crypto '88.
13. I. Damgård and J.B. Nielsen. Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. Crypto 2002.
14. C. Dwork and M. Naor. Zaps and Their Applications. FOCS 2000.
15. U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. Crypto '89.
16. M. Fischlin. Round-Optimal Composable Blind Signatures in the Common Reference String Model. Crypto 2006.
17. O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools.* Cambridge University Press, 2001.
18. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM J. Computing* 17(2): 281–308 (1988).
19. A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. Crypto '97.
20. Y. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent General Composition of Secure Protocols in the Timing Model. STOC 2005.
21. A. Kiayias and H.-S. Zhou. Two-Round Concurrent Blind Signatures without Random Oracles. SCN 2006.

22. Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *J. Cryptology* 16(3): 143–184 (2003).
23. Y. Lindell. Bounded-Concurrent Secure Two-Party Computation without Setup Assumptions. STOC 2003.
24. Y. Lindell. Lower Bounds for Concurrent Self-Composition. TCC 2004.
25. S. Micali, R. Pass, and A. Rosen. Input-Indistinguishable Computation. FOCS 2006.
26. T. Okamoto. Efficient Blind and Partially Blind Signatures without Random Oracles. TCC 2006.
27. D. Pointcheval. Strengthened Security for Blind Signatures. Eurocrypt '98.
28. D. Pointcheval and J. Stern. Provably Secure Blind Signature Schemes. Asiacrypt '96.
29. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology* 13(3): 361–396 (2000).
30. M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. FOCS 2002.
31. M. Prabhakaran and A. Sahai. New Notions of Security: Achieving Universal Composability without Trusted Setup. STOC 2004.
32. A. Rosen. *Concurrent Zero-Knowledge.* Springer, 2006.

A ZAPs

A ZAP is a 2-round witness-indistinguishable proof (with negligible soundness error) for some $\mathcal{NP}$-language L with associated relation R_L. Formally, let $L_{p(k)} \stackrel{\text{def}}{=} L \cap \{0,1\}^{\leq p(k)}$. A ZAP consists of two polynomial-time interactive algorithms $\mathcal{P}, \mathcal{V}$ along with a polynomial p such that:

- On input 1^k, the verifier $\mathcal{V}$ outputs an initial message ρ.
- On input ρ, a statement $x \in L_{p(k)}$, and a witness w such that $(x, w) \in R_L$, the prover $\mathcal{P}$ outputs a proof π.
- Given ρ, x, and π, the verifier $\mathcal{V}$ outputs a decision bit.

For any k and (x, w) as above, $\mathcal{V}(\rho, x, \mathcal{P}(\rho, x, w)) = 1$ with probability 1. A ZAP also satisfies *adaptive soundness* even against all-powerful cheating provers; that is, for arbitrary $\mathcal{P}^*$ the following is negligible:

$$\Pr\left[\rho \leftarrow \mathcal{V}(1^k); (x, \pi) \leftarrow \mathcal{P}^*(\rho) : \mathcal{V}(\rho, x, \pi) = 1 \bigwedge x \notin L\right].$$

We define witness indistinguishability by requiring that the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:

1. $\mathcal{A}(1^k)$ outputs a string ρ, a sequence $x_1, \ldots, x_\ell \in L_{p(k)}$, and two sequences $w_1^0, \ldots, w_\ell^0$ and $w_1^1, \ldots, w_\ell^1$. It is required that $(x_i, w_i^0), (x_i, w_i^1) \in R_L$ for all i.
2. A random bit b is chosen.
3. Compute $\pi_i \leftarrow \mathcal{P}(\rho, x_i, w_i^b)$ for all i, and give these to $\mathcal{A}$.
4. $\mathcal{A}$ outputs a bit b'. The advantage of $\mathcal{A}$ is $\left|\Pr[b' = b] - \frac{1}{2}\right|$.

ZAPs can be constructed based on trapdoor permutations [14].