A preliminary version appears in Advances in Cryptology — Crypto 2006, Lecture Notes in Computer Science, Springer-Verlag.

# Round-Optimal Composable Blind Signatures in the Common Reference String Model

Marc Fischlin*

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

**Abstract** We build concurrently executable blind signatures schemes in the common reference string model, based on general complexity assumptions, and with optimal round complexity. Namely, each interactive signature generation requires the requesting user and the issuing bank to transmit only one message each. We also put forward the definition of universally composable blind signature schemes, and show how to extend our concurrently executable blind signature protocol to derive such universally composable schemes in the common reference string model under general assumptions. While this protocol then guarantees very strong security properties when executed within larger protocols, it still supports signature generation in two moves.

## 1 Introduction

Blind signatures, introduced by Chaum [Cha83], allow a bank to interactively issue signatures to users such that the signed message is hidden from the bank (blindness) while at the same time users cannot output more signatures than interactions with the bank took place (unforgeability). Numerous blind signature schemes have been proposed, mostly under specific number-theoretic assumptions, some relying also on the random oracle model [PS00,Abe01,BNPS03,Bol03] and some forgoing random oracles [CKW04,KZ05,Oka06]. Only the work by Juels et al. [JLO97] addresses the construction of blind signatures under general assumptions explicitly, and deploys general two-party protocols and oblivious transfer based on trapdoor permutations.

Interestingly, almost all of the aforementioned blind signature schemes require three or more moves (most of them even in the random oracle model) and concurrent executions of the signature generation protocol are often a concern (cf. [Poi98,PS00,Oka06]). For instance, making the solution by Juels et al. [JLO97] concurrently secure would require a high round complexity. This follows from results by Lindell [Lin03,Lin04] showing, among others, that in the plain model the number of rounds in blind signature schemes with black-box security proofs is bounded from below by the number of concurrent executions.

A notable exception to the problems with interleaving executions are schemes with an optimal two-move signature generation protocol, solving the concurrency problem immediately. This includes Chaum's original RSA-based blind signature protocol and the pairing-based discrete-log version thereof [Bol03]. Unfortunately, the security proofs [BNPS03,Bol03] for these schemes need the random oracle model —by which they can bypass Lindell's lower bound for the plain model— and rely on the so-called one-more RSA or one-more discrete-log assumptions, which are less investigated than the corresponding standard problems.

* This work was supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

Here we show that one can build secure blind signature schemes with a two-move signature generation protocol under general assumptions (namely, trapdoor permutations and, depending on the level of unforgeability, also collision-resistant hash functions). Our scheme does not rely on random oracles, yet to bridge the lower bound on the number of rounds we work in the common reference string model. We note that instead of falling back on general multi-party paradigms as in [JLO97] and inheriting the round complexity of the underlying oblivious transfer protocols for general assumptions we give a dedicated solution to the blind signature problem.

*Construction Idea.* The basic (and simplified) construction idea of our blind signature scheme is as follows. The user commits to the message $m$ and sends this commitment $U$ to the bank. The bank signs $U$ with a (strongly unforgeable) signature scheme and sends the signature $B$ back to user. The user finally derives the blind signature for $m$ by computing a commitment[1] $C$ of $U||B$ and proving with a non-interactive zero-knowledge proof $\pi$ (based on the common reference string) that this commitment $C$ contains a valid signature $B$ for $U$ and that $U$ itself is a commitment of the message $m$. The blind signature to $m$ is given by the commitment $C$ and the proof $\pi$.

Using standard non-interactive zero-knowledge (NIZK) proofs our protocol above would provide a weaker unforgeability notion than postulated in [PS00,JLO97]. That is, a malicious user could easily generate several signatures from a single interaction with the bank by generating multiple proofs $\pi_1, \pi_2, \ldots$ for the same commitment $C$. All these pairs $C||\pi_i$ would be valid blind signatures for the same message $m$, while the standard unforgeability definition asks that a malicious user cannot create more signatures than interactions happened.

Fortunately, we can indeed turn the above idea into a scheme supporting the desired unforgeability level by using so-called *unique* non-interactive zero-knowledge (UNIZK) proofs, recently introduced by Lepinski et al. [LMS05] and shown to be realizable under the quadratic residuosity assumption. For such unique proof systems there is a bijection between the witnesses $w$ to a theorem $x$ and the valid zero-knowledge proofs $\pi$, guaranteeing together with the uniqueness of the commitment $C$ that there is only a single valid proof for each signature interaction.

Here we only need a relaxation to so-called *single-theorem* unique zero-knowledge (sUNIZK) proofs, in which uniqueness and zero-knowledge hold simultaneously for one theorem only (but each property individually remains true for an unbounded sequence of theorems), As briefly mentioned in [LMS05], this relaxation enables constructions of such proofs for 3SAT based on trapdoor permutations, and here we show how to build such proofs under the same assumption for CircuitSAT. Since we deploy such sUNIZK proofs mainly as a tool to achieve the strong security guarantees of our blind signature protocol, most of the details about sUNIZK proofs are delegated to the appendix.

*Universal Composition.* As explained, our two-move signature generation protocol overcomes the concurrency problem effortlessly. More general a slight variation of our scheme yields a secure blind signature scheme in the universal composition (UC) framework [Can01]. Secure schemes in this UC framework enable interdependent executions with other protocols while preserving the main security characteristics. Our modified scheme now requires the same assumptions as before as well as a simulation-sound NIZK proof of knowledge, which can also be derived from trapdoor permutations [Sah99,DDO$^+$01].

[1] For the security proof we require that the commitment $C$ is actually done through an encryption scheme.

Towards proving the universal composability result we first formalize an ideal functionality $\mathcal{F}_{\text{BlSig}}$, prescinding the basic requirements of blind signatures such as completeness, unforgeability and blindness. Since such UC blind signatures can be used to build UC commitments it follows from an impossibility result in [CF01] that UC blind signatures cannot be realized through two-party protocols in the plain model. But, as our solution shows, augmenting the model by common reference strings one can build UC blind signatures (against non-adaptive corruptions) under general assumptions. Compared to general feasibility results in the UC framework [CLOS02], showing that essentially any functionality can be securely realized in the common random string, our solution still needs only two-moves to generate signatures.

## 2 Blind Signatures in the Common Reference String Model

In this section we briefly introduce the underlying tool, single-theorem UNIZK proofs, and their construction for CircuitSAT; more details can be found in Appendix A. We then recall the security definition of blind signatures and finally present our two-move solution and prove its security.

### 2.1 Single-Theorem UNIZK Proofs

Lepinski et al. [LMS05] recently introduced the notion of *unique* non-interactive zero-knowledge (UNIZK) proofs. For such unique proof systems the prover in mode key first generates a public-key $PK$ and then uses the corresponding secret key in mode prove to generate zero-knowledge proofs $\pi_i$ for an unbounded (but polynomial) number of subsequently chosen statements $x_i$ and witnesses $w_i$. Uniqueness says that, once the key $PK$ has been chosen, for each statement $x_i$ there is exactly one valid proof $\pi_i$ per witness $w_i$.

Here we consider *single-theorem* unique zero-knowledge (sUNIZK) proofs, briefly touched in [LMS05], where the prover's public key $PK$ allows to prove exactly one theorem $x$ such that the proof is zero-knowledge *and* unique at the same time. Uniqueness still holds if several statements for the same key $PK$ are proven but then the proofs may not be zero-knowledge anymore. Vice versa, the system should still be multiple zero-knowledge in the sense of [FLS99], i.e., one can proof several statements $x_i$ in zero-knowledge *for the same common random string*, but then the prover must choose a new key $PK_i$ for each proof (and each proof is individually unique with respect to $PK_i$).

Single-theorem UNIZK proofs obey first of all the three basic properties of regular NIZK proofs: completeness, soundness and (unbounded) zero-knowledge. We omit a formal definition of these properties as they are standard. To formalize the (single-theorem) uniqueness we follow [LMS05] and define uniqueness by a bijection between witnesses and valid proofs (if any). To this end we parameterize the underlying $\mathcal{NP}$ relation $R$ by the common reference (or random) string $crs$, generated by an algorithm $\mathcal{C}$, and the public key $PK$. These parameters are both themselves determined according to a complexity parameter $n$, and for such parameters $R_{crs,PK}$ takes as inputs $x \in \{0,1\}^{\chi(n)}$ and $w \in \{0,1\}^{\omega(n)}$, where $x, w$ may depend on $crs, PK$. Let $W_{crs,PK}(x) = \left\{ w \in \{0,1\}^{\omega(n)} \;\middle|\; R_{crs,PK}(x,w) = 1 \right\}$ denote the set of witnesses to $x$ with respect to $crs$ and $PK$.

**Definition 1 (Single-Theorem Unique Zero-Knowledge).** *A NIZK proof $(\mathcal{C}, \mathcal{P}, \mathcal{V})$ for relation $R$ is a single-theorem unique NIZK proof system if the following holds:*

**Uniqueness.** *With overwhelming probability over the choice of $crs \leftarrow \mathcal{C}(1^n)$, for any $PK$ and any $x$, if the set $\Pi_{crs,PK}(x) = \{\pi \mid \mathcal{V}(crs, x, PK, \pi) = 1\}$ of accepted proofs in not empty,*

*then there exists a bijection $\tau_{crs,PK,x}$ between the set $\Pi_{crs,PK}(x)$ and the set $W_{crs,PK}(x) = \{w \mid (x,w) \in R_{crs,PK}\}$ of witnesses.*

As pointed out in [LMS05] such sUNIZK proofs are easier to construct than general UNIZK proofs, and an appropriate modification of the NIZK protocol with preprocessing in [DMP88] gives a solution for 3SAT which obeys single-theorem uniqueness under general assumptions. Our construction instead follows the one by Damgård [Dam93] for CircuitSATwhere one should decide for a given circuit $\mathsf{C}_n$ and value $x \in \{0,1\}^{\chi(n)}$ whether a satisfying input $w \in \{0,1\}^{\omega(n)}$ for $\mathsf{C}_n(x,\cdot)$ exist. The advantage of CircuitSAT over 3SAT, pointed out in [Dam93], is that it allows to prove $\mathcal{NP}$ statements directly without reductions to 3SAT. Yet, compared to the solution in [Dam93] using the quadratic residuosity assumption to derive possibly ambiguous proofs, our goal is to use general assumptions and to achieve uniqueness. Details can be found in Appendix A; here we merely state the main result:

**Proposition 1.** *Let $\mathcal{C} = (\mathcal{C}_n)_{n\in\mathbb{N}}$ be a sequence of circuits $\mathcal{C}_n : \{0,1\}^{\chi(n)} \times \{0,1\}^{\omega(n)} \to \{0,1\}$. Single-theorem unique non-interactive zero-knowledge proof systems for relation $R^{\mathcal{C}} = \left\{(x,w) \in \{0,1\}^{\chi(n)+\omega(n)} \;\middle|\; \mathcal{C}_n(x,w) = 1\right\}$ in the common random string model exist if (regular) non-interactive zero-knowledge proof systems for $\mathcal{NP}$ in the common random string model and one-way permutations exist.*

The proposition holds for both bounded and unbounded provers. For efficient provers it follows that sUNIZK proofs for CircuitSAT exist if trapdoor permutations exist, and for unbounded provers they can be built from any one-way permutation [FLS99].

### 2.2 Blind Signatures and Their Security

For the interactive signature generation protocol of a blind signature scheme we introduce the following notation. For two interactive algorithms $\mathcal{X}$, $\mathcal{Y}$ we denote by $(a,b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y)\rangle$ the joint execution of $\mathcal{X}$ for input $x$ and $\mathcal{Y}$ for input $y$, where $\mathcal{X}$'s private output at the end of the execution equals $a$ and $\mathcal{Y}$'s private output is $b$. For an algorithm $\mathcal{Y}$ we write $\mathcal{Y}^{\langle \mathcal{X}(x),\cdot\rangle^\infty}$ if $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$ in arbitrarily interleaved order. Accordingly, $\mathcal{X}^{\langle\cdot,\mathcal{Y}(y_0)\rangle^1,\langle\cdot,\mathcal{Y}(y_1)\rangle^1}$ can invoke arbitrarily interleaved executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$ but interact with each algorithm only once.

**Definition 2 (Blind Signature Scheme).** *A blind signature scheme (in the common reference string model) consists of a tuple of efficient algorithms $\mathsf{BS} = (\mathcal{C}_{BS}, \mathsf{KG}_{BS}, \langle \mathcal{B}, \mathcal{U}\rangle, \mathsf{Vf}_{BS})$ where*

**CRS generation.** *$\mathcal{C}_{BS}$ on input $1^n$ outputs a common reference (or random) string $crs_{BS}$.*

**Key Generation.** *$\mathsf{KG}_{BS}(crs_{BS})$ generates a key pair $(sk_{BS}, pk_{BS})$.*

**Signature Issuing.** *The joint execution of algorithms $\mathcal{B}(crs_{BS}, sk_{BS})$ and $\mathcal{U}(crs_{BS}, pk_{BS}, m)$ generates an output $S$ of the user, $(\perp, S) \leftarrow \langle \mathcal{B}(crs_{BS}, sk_{BS}), \mathcal{U}(crs_{BS}, pk_{BS}, m)\rangle$.*

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

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

Security of blind signatures consists of two requirements [PS00,JLO97]. Unforgeability says that it should be infeasible for a malicious user $\mathcal{U}^*$ to generate $k+1$ valid signatures

given that $k$ interactions with the honest bank took place (where the adversary adaptively decides on the number $k$ of interactions). Blindness says that it should be infeasible for a malicious bank $\mathcal{B}^*$ to determine the order in which two messages $m_0, m_1$ have been signed in executions with an honest user. This should hold, of course, as long as the interactive signature generation produces two valid signatures and the bank $\mathcal{B}^*$ for example does not abort deliberately in one of the two executions.

**Definition 3 (Secure Blind Signature Scheme).** *A blind signature scheme* $\mathsf{BS} = (\mathcal{C}_{BS}, \mathsf{KG}_{BS}, \langle \mathcal{B}, \mathcal{U} \rangle, \mathsf{Vf}_{BS})$ *in the common reference string model is called secure if the following holds:*

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

***Experiment*** $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}(n)$
$crs_{BS} \leftarrow \mathcal{C}_{BS}(1^n)$
$(sk_{BS}, pk_{BS}) \leftarrow \mathsf{KG}_{BS}(crs_{BS})$
$((m_1, S_1), \ldots, (m_{k+1}, S_{k+1})) \leftarrow \mathcal{U}^{*\langle \mathcal{B}(crs_{BS}, sk_{BS}), \cdot \rangle^{\infty}}(crs_{BS}, pk_{BS})$
*Return* 1 *iff*
$(m_i, S_i) \neq (m_j, S_j)$ *for* $1 \leq i < j \leq k+1$, *and*
$\mathsf{Vf}_{BS}(crs_{BS}, pk_{BS}, m_i, S_i) = 1$ *for all* $i = 1, 2, \ldots, k+1$, *and*
*at most* $k$ *interactions with* $\langle \mathcal{B}(crs_{BS}, sk_{BS}), \cdot \rangle^{\infty}$ *were initiated.*

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

***Experiment*** $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{B}^*}(n)$
$crs_{BS} \leftarrow \mathcal{C}_{BS}(1^n)$
$(pk_{BS}, m_0, m_1, \beta_{\mathsf{find}}) \leftarrow \mathcal{B}^*(\mathsf{find}, crs_{BS})$
$b \leftarrow \{0, 1\}$
$\beta_{\mathsf{issue}} \leftarrow \mathcal{B}^{*\langle \cdot, \mathcal{U}(crs_{BS}, pk_{BS}, m_b) \rangle^1, \langle \cdot, \mathcal{U}(crs_{BS}, pk_{BS}, m_{1-b}) \rangle^1}(\mathsf{issue}, \beta_{\mathsf{find}})$
*and let* $S_b, S_{1-b}$ *denote the (possibly undefined) local outputs of* $\mathcal{U}(crs_{BS}, pk_{BS}, m_b)$ *resp.* $\mathcal{U}(crs_{BS}, pk_{BS}, m_{1-b})$.
$b^* \leftarrow \mathcal{B}^*(\mathsf{guess}, S_0, S_1, \beta_{\mathsf{issue}})$
*Return a random bit if* $S_0 = \perp$ *or* $S_1 = \perp$*, else return* 1 *iff* $b = b^*$.

In a *partially* blind signature scheme [AF96] the bank and the user first agree on some information $\mathsf{info}$ which is attached to the blind signature. There, unforgeability demands that a malicious user cannot find $k+1$ distinct but valid tuples $(\mathsf{info}_i, m_i, S_i)$. The definition of blindness then allows a malicious bank to determine $\mathsf{info}$ together with the messages $m_0, m_1$ such that the bank still cannot decide the order in which the users execute the issuing protocol for $\mathsf{info}, m_0$ and $\mathsf{info}, m_1$, respectively. Jumping ahead we note that we can easily turn our blind signature scheme into a partially blind one.

In a weaker unforgeability requirement we only demand that the messages $m_1, \ldots, m_{k+1}$ output by the malicious user are distinct (instead of stipulating distinct message-signature pairs). Clearly, the unforgeability notion in Definition 3 implies this weaker requirement. For our scheme this relaxation also allows to dismiss unique zero-knowledge proofs and to use regular NIZK proofs instead.

### 2.3 Construction

The high-level idea of our blind signature protocol is as follows. To obtain a blind signature from the bank the user commits to the message $m$ and sends this commitment $U$ to the bank (where the commitment has the additional property of unique randomness). The bank signs the commitment with a regular but strongly unforgeable signature scheme $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$ and returns the signature $B$ to the user. The user derives the blind signature for $m$ by computing another commitment $C$ of the signature $B$ together with $U$, and a unique non-interactive zero-knowledge proof $\pi$ showing the validity of $C$.

We have already remarked that single-theorem UNIZK proofs are sufficient for our purpose. This is because we can let the user generate a new key $PK$ at the beginning of each interaction and have $PK$ included in the commitment $U$. The twist here is that we use a hash function $H$ from a family $\mathcal{H}$ to hash down the key $PK$ to a short string of length $h(n)$. This is necessary as we later use $PK$ in the statement $x$ for the zero-knowledge proof and because the length of $PK$ is determined as a function of the length of $x$ itself. Using the $h(n)$-bit hash value overcomes this dependency.

For the same reason as in the case of $PK$ we also assume that the signature scheme is *length-invariant*, i.e., that public keys $pk_{\mathsf{Sig}}$ as well as signatures for security parameter $n$ are all of the same length $s(n)$. We note that this can always be achieved by standard padding techniques, and strongly unforgeable, length-invariant signature schemes exist if one-way functions exist [NY89,Rom90,Gol04].

We furthermore assume that the commitment scheme $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ in the common random string model, given by algorithms $\mathcal{C}_{\mathsf{Com}}$ generating the string $crs_{\mathsf{Com}}$ and algorithm $\mathsf{Com}(crs_{\mathsf{Com}}, \cdot, \cdot) : \{0,1\}^{2n+h(n)} \times \{0,1\}^n \to \{0,1\}^{c(n)}$ mapping strings from $\{0,1\}^{2n+h(n)}$ with $n$-bit randomness to commitments, is length-invariant, too. That is, the reference strings as well as commitments are all of length $c(n)$ for parameter $n$. We also need that the commitment scheme has *unique openings*, i.e., with overwhelming probability over the choice of $crs_{\mathsf{Com}} \leftarrow \mathcal{C}(1^n)$ there do not exist $(z, r) \neq (z', r')$ with $\mathsf{Com}(crs_{\mathsf{Com}}, z; r) = \mathsf{Com}(crs_{\mathsf{Com}}, z'; r')$. Such commitment schemes can be derived for instance from one-way permutation based pseudorandom generators.

In order to turn the above idea into a provably secure scheme the proof $\pi$ needs to allow extraction of $U$ and $B$ encapsulated in $C$. We accomplish this by using an IND-CPA secure encryption scheme $(\mathsf{KG}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ to "commit" to $U||B$ in $C$, where the public key $pk_{\mathsf{Enc}}$ of the encryption algorithm becomes part of the common reference string.[2] We presume that the encryption scheme is also length-invariant and that public keys $pk_{\mathsf{Enc}}$ and ciphertexts $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$ for $U||B \in \{0,1\}^{c(n)+s(n)}$ and randomness $v \in \{0,1\}^n$ are all of length $e(n)$. This is again without loss of generality.

The circuit $\mathsf{C}_n^{\mathrm{BS}}$ for parameter $n$ we consider for the sUNIZK proof takes as input strings $x = C||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK)||m$ of bit length $\chi(n) = c(n) + 2e(n) + h(n) + s(n) + n$ and $w = u||v||B$ of length $\omega(n) = 2n + s(n)$, and returns an output bit which is determined as follows. The circuit is built from the descriptions of algorithms $\mathsf{Com}, \mathsf{Enc}, \mathsf{Vf}_{\mathsf{Sig}}$ and checks for the signature's verification algorithm that $\mathsf{Vf}_{\mathsf{Sig}}(pk_{\mathsf{Sig}}, \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u), B) = 1$ and that the value $C$ equals the ciphertext $\mathsf{Enc}(pk_{\mathsf{Enc}}, \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)||B; v)$. If and only if both tests evaluate to true then the circuit outputs 1. The corresponding relation is given by $R^{\mathrm{BS}}_{crs,PK} = \left\{(x, w) \in \{0,1\}^{\chi(n)} \times \{0,1\}^{\omega(n)} \;\middle|\; \mathsf{C}_n^{\mathrm{BS}}(x, w) = 1\right\}$.

[2] We can also assume that we have dense public-key schemes [DP92] and that the common *random* string contains such a public key.

**Construction 1 (Blind Signature Scheme).** *Let* $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$ *be a signature scheme,* $(\mathsf{KG}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme,* $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ *be a commitment scheme,* $\mathcal{H}$ *be a hash function family and let* $(\mathcal{C}_{uni}, \mathcal{P}, \mathcal{V})$ *be a non-interactive zero-knowledge proof system for* $R^{BS}$*. Define the following four procedures:*

**CRS Generation.** *Algorithm* $\mathcal{C}_{BS}(1^n)$ *runs* $crs_{uni} \leftarrow \mathcal{C}_{uni}(1^n)$*,* $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$ *and* $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}) \leftarrow \mathsf{KG}_{\mathsf{Enc}}(1^n)$*. It outputs* $crs_{BS} \leftarrow (crs_{uni}, crs_{\mathsf{Com}}, pk_{\mathsf{Enc}})$*.*

**Key Generation.** *The bank's key generation algorithm* $\mathsf{KG}_{BS}(crs_{BS})$ *generates a signature key pair* $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) \leftarrow \mathsf{KG}_{\mathsf{Sig}}(1^n)$ *and also picks a hash function* $H \leftarrow \mathcal{H}(1^n)$*. It returns* $(pk_{BS}, sk_{BS}) \leftarrow ((pk_{\mathsf{Sig}}, H), sk_{\mathsf{Sig}})$*.*

**Signature Issue Protocol.** *The interactive signature issue protocol is described in Figure 1.*

**Signature Verification.** *The verification algorithm* $\mathsf{Vf}_{BS}(crs_{BS}, m, S)$ *parses the signature* $S$ *as* $S = C||PK||\pi$ *and returns the output* $\mathcal{V}_{uni}(crs_{uni}, PK, x, \pi)$ *for the value* $x = C||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK)||m$*.*

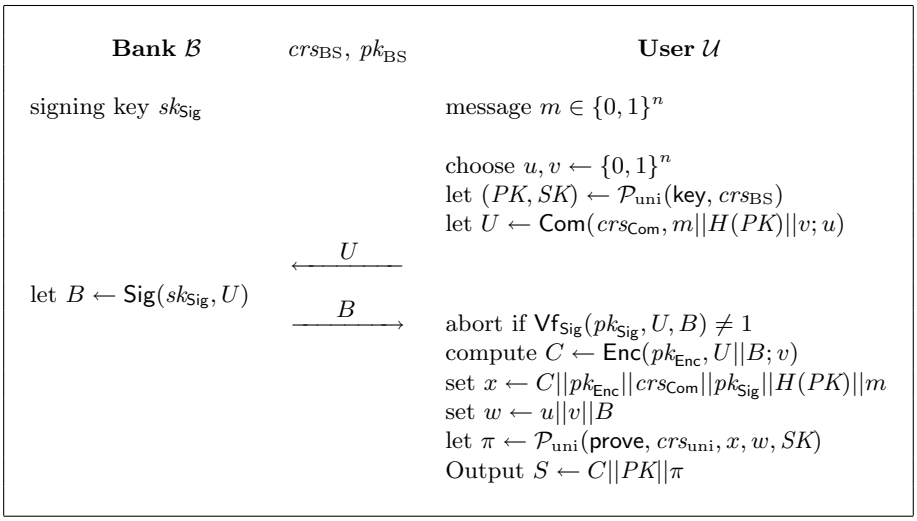


**Figure 1.** Blind Signature Scheme: Issue Protocol

**Theorem 2.** *Let* $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$ *be a length-invariant signature scheme which is strongly unforgeable against adaptive chosen-message attacks,* $(\mathsf{KG}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be a length-invariant IND-CPA secure encryption scheme,* $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ *be a length-invariant non-interactive commitment scheme with unique openings in the common random string model, and* $\mathcal{H}$ *be a collision-intractable hash function family. Let* $(\mathcal{C}_{uni}, \mathcal{P}_{uni}, \mathcal{V}_{uni})$ *be a single-theorem unique non-interactive zero-knowledge proof system for* $R^{BS}$*. Then the scheme defined in Construction 1 is a secure blind signature scheme.*

*Proof.* We first show unforgeability and then blindness.

*Unforgeability.* Assume that there exists an adversary $\mathcal{U}^*$ such that with noticeable probability the following holds. On input $crs_{\mathrm{BS}}, pk_{\mathrm{BS}}$ the adversary $\mathcal{U}^*$ manages to output $k+1$ valid signatures $S_i = C_i||PK_i||\pi_i$ for messages $m_i$ after at most $k$ interactions with the honest bank $\mathcal{B}$ (and where the pairs $(m_i, S_i)$ are pairwise distinct). Given such an adversary we construct a successful adversary $\mathcal{A}$ against the security of the signature scheme $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$.

Adversary $\mathcal{A}$ is given as input a public key $pk_{\mathsf{Sig}}$ of the signature scheme and is granted oracle access to a signature oracle $\mathsf{Sig}(sk_{\mathsf{Sig}}, \cdot)$. This adversary first generates $crs_{\mathrm{uni}} \leftarrow \mathcal{C}_{\mathrm{uni}}(1^n)$ for the zero-knowledge proof, $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$ for the commitments, picks a hash function $H \leftarrow \mathcal{H}(1^n)$ and $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}) \leftarrow \mathsf{KG}_{\mathsf{Enc}}(1^n)$ and sets $crs_{\mathrm{BS}} \leftarrow (crs_{\mathrm{uni}}, crs_{\mathsf{Com}}, pk_{\mathsf{Enc}})$. It next invokes a black-box simulation of $\mathcal{U}^*$ for input $(crs_{\mathrm{BS}}, (pk_{\mathsf{Sig}}, H))$. Each time the user $\mathcal{U}^*$ initiates the issue protocol with the bank, algorithm $\mathcal{A}$ uses the signature oracle to answer the request $U$ by a signature $B \leftarrow \mathsf{Sig}(sk_{\mathsf{Sig}}, U)$. When the adversary finally outputs the message/signature pairs $(m_i, S_i)$ algorithm $\mathcal{A}$ parses each $S_i$ as $S_i = C_i||PK_i||\pi_i$ and decrypts each $C_i$ to $U_i||B_i$. Algorithm $\mathcal{A}$ outputs the first one of these pairs $(U_i, B_i)$ which has not been the result of one of the interactions with the signing oracle (or returns $\perp$ if no such pair exist).

For the analysis assume that $\mathcal{U}^*$ succeeds with noticeable probability. Since $k+1$ pairs $(m_i, S_i)$ are valid, all the proofs $\pi_i$ in $S_i = C_i||PK_i||\pi_i$ are valid as well. Hence, with overwhelming probability over the choice of $crs_{\mathrm{uni}}$ each $C_i$ is a valid ciphertext of some $U_i||B_i$ under $pk_{\mathsf{Enc}}$, and each $U_i$ commits to the corresponding message $m_i$ and to the hash value of $PK_i$. Moreover, $U_i$ also commits to randomness $v_i$ with which $U_i||B_i$ is encrypted in $C_i$.

Suppose that there exist $(U_i, B_i) = (U_j, B_j)$ for $i \neq j$ but such that the two distinct message/signature pairs $(m_i, C_i||PK_i||\pi_i)$ and $(m_j, C_j||PK_j||\pi_j)$ are valid. But then the values $U_i, U_j$ specify the same unique tuple $(m_i, H(PK_i), v_i) = (m_j, H(PK_j), v_j)$ and, because of unique openings, also the same randomness $u_i = u_j$, and thus $C_i = C_j$. We can furthermore assume that no collisions $H(PK_i) = H(PK_j)$ for $PK_i \neq PK_i$ for any $i, j$ occur, else this would contradict the collision-resistance of $\mathcal{H}$. Therefore, $PK_i = PK_j$ and the only place where these message/signature pairs can differ is for $\pi_i$ and $\pi_j$. But by the uniqueness property of the zero-knowledge proof we conclude that there cannot exist valid proofs $\pi_i \neq \pi_j$ for the same theorem

$$x = C_i||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK_i)||m_i = C_j||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK_j)||m_j$$

and the same witness

$$w = u_i||v_i||B_i = u_j||v_j||B_j,$$

with overwhelming probability over the choice of the common random string $crs_{\mathrm{uni}}$. It follows that no such pairs $(U_i, B_i) = (U_j, B_j)$ for $i \neq j$ can exist (except for negligible probability).

Given that all $(U_i, B_i)$ are distinct, we can sort out those among these pairs where the bank has signed $U_i$ with $B_i$. Because of the uniqueness of the pairs there must still be one $U_i$ left which has not been signed by the bank with $B_i$. Hence, with noticeable probability adversary $\mathcal{A}$ produces a (strong) forgery.

*Blindness.* To prove blindness we consider an adversarial controlled bank $\mathcal{B}^*$ in experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{B}^*}(n)$. We gradually transform the way the signatures $S_0 = C_0||PK_0||\pi_0$ and $S_1 = C_1||PK_1||\pi_1$ are computed such that, at then end, they are completely independent of bit $b$.

In the first step we replace all the steps involving the prover by the output of the zero-knowledge simulator. That is, consider the following modified procedures of the blind signature scheme (key generation and verification remain unchanged):

**CRS Generation.** Algorithm $\mathcal{C}_{\mathrm{BS}}(1^n)$ runs $(crs_{\mathrm{uni}}, \sigma) \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{crs}, 1^n)$, $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$ and $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}) \leftarrow \mathsf{KG}_{\mathsf{Enc}}(1^n)$. It outputs $crs_{\mathrm{BS}} \leftarrow (crs_{\mathrm{uni}}, crs_{\mathsf{Com}}, pk_{\mathsf{Enc}})$.

**Signature Issue Protocol.** For the signature issuing the user now also picks $u, v \leftarrow \{0,1\}^n$, but lets $(PK, SK) \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{key}, \sigma)$ instead. The user again sends the commitment $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)$ to the bank $\mathcal{B}^*$ which replies with some $B$. The user aborts if $\mathsf{Vf}_{\mathsf{Sig}}(pk_{\mathsf{Sig}}, U, B) \neq 1$ and else computes $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$ as well as $\pi \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{prove}, \sigma, x, SK)$. Output $S \leftarrow C||PK||\pi$.

Denote the modified scheme by $\mathsf{BS}'$. It follows easily from the zero-knowledge property that experiments $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{B}^*}(n)$ and $\mathsf{Blind}^{\mathsf{BS}'}_{\mathcal{B}^*}(n)$ return 1 with the same probability (except for a negligible probability).

In the next step we further modify the signature scheme by replacing the commitments $U$ by commitments to all-zero strings. More precisely, we change the signature issue protocol of the blind signature scheme $\mathsf{BS}'$ as follows (recall that this modified scheme already uses the zero-knowledge simulator to prepare the signatures):

**Signature Issue Protocol.** The user picks $u, v \leftarrow \{0,1\}^n$, sets $(PK, SK) \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{key}, \sigma)$ as before, but now sends $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, 0^{2n+h(n)}; u)$ to the bank to get a signature $B$ (which is also checked). It then computes again $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$, $\pi \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{prove}, \sigma, x, SK)$ and outputs $S \leftarrow C||PK||\pi$.

We call this modified scheme $\mathsf{BS}''$. It is easy to see that, by the secrecy of the commitment scheme, the difference in the output distributions of experiments $\mathsf{Blind}^{\mathsf{BS}'}_{\mathcal{B}^*}(n)$ and $\mathsf{Blind}^{\mathsf{BS}''}_{\mathcal{B}^*}(n)$ is negligible.

Finally, we replace the encryption of $U$ and $B$ by an encryption of the all-zero string, i.e., we modify the signature issuing protocol from $\mathsf{BS}''$ further to obtain:

**Signature Issue Protocol.** The user selects $u, v \leftarrow \{0,1\}^n$ as before and again computes $(PK, SK) \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{key}, \sigma)$ and $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, 0^{2n+h(n)}; u)$. For the bank's reply $B$ to $U$ it checks validity and this time sets $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, 0^{c(n)+s(n)}; v)$, $\pi \leftarrow \mathcal{Z}_{\mathrm{uni}}(\mathsf{prove}, \sigma, x, SK)$ and outputs $S \leftarrow C||PK||\pi$.

We call this modified scheme by $\mathsf{BS}'''$. By the IND-CPA security of the encryption scheme we conclude that the difference in the output distributions of experiments $\mathsf{Blind}^{\mathsf{BS}''}_{\mathcal{B}^*}(n)$ and $\mathsf{Blind}^{\mathsf{BS}'''}_{\mathcal{B}^*}(n)$ is also negligible.

In experiment $\mathsf{Blind}^{\mathsf{BS}'''}_{\mathcal{B}^*}(n)$ the signatures $S = C||PK||\pi$ are independent of the data $U, B$ in the signature generation protocol. Hence, the adversary $\mathcal{B}^*$ cannot predict $b$ better than with probability $1/2$. Conclusively, the probability of the original experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{B}^*}(n)$ to return 1 must be negligibly close to $1/2$, proving blindness. □

To get a partially blind signature scheme, where the signer and the user share some public information $\mathsf{info}$ to be included in the blind signature, we let the bank simply sign the user's commitment $U$ together with $\mathsf{info}$, i.e., $B \leftarrow \mathsf{Sig}(sk, \mathsf{info}||U)$ and also let the user base the correctness proof $\pi$ on $\mathsf{info}$. The security proof carries over straightforwardly.

Also, to get a weakly unforgeable blind signature scheme where the malicious user only succeeds when outputting $k+1$ distinct messages we can drop the requirement on the strong unforgeability of the bank's signature scheme and use a regular signature scheme guaranteeing basic unforgeability only, and we can also dismiss the requirement on unique randomness for the commitment scheme and merely need statistically-binding commitments. Furthermore,

we can then also switch to regular (not necessarily unique) zero-knowledge proofs. The latter implies that the user does not have to pick a public key $PK$ before the first move and that one may think of $PK$ as being the empty string. Consequently, the hash function becomes obsolete and we get such blind signatures under the sole assumption that trapdoor permutations exist.

## 3 Universally Composable Blind Signatures

As mentioned, the blind signature scheme in the previous section allows concurrent executions of the signature generation protocol, i.e., when the protocol is interleaved with itself. More generally, one would like to have a guarantee that such a scheme still supports the basic security properties, even when run as a building block within larger protocols, independently how the execution is intertwined with other steps. Such a guarantee is provided by the universal composition (UC) framework [Can01].

In the UC framework one defines an idealized version of the primitive in question, capturing the desired security properties in an abstract way and ensuring that the functionality is secure in interdependent settings. Given an appropriate formalization of some functionality $\mathcal{F}$ in the UC framework, one next shows that this functionality can be securely realized by an interactive protocol between the parties (without the trusted interface). Here, securely realizing means that, in any environment (modeled through an algorithm $\mathcal{Z}$) in which the protocol may be run, for this environment executions of the interactive protocol in presence of an adversary $\mathcal{A}$ are indistinguishable from executions in the ideal model with the trustworthy functionality $\mathcal{F}$ and an ideal-model adversary $\mathcal{S}$. The UC framework, notably the composition theorem, then ensures that the protocol can indeed be securely deployed as a subroutine in more complex protocols and environments.

A formal introduction to the UC framework is beyond the scope of our paper; we refer to [Can01] to a comprehensive definition. We remark that we consider non-adaptive adversaries here which corrupt parties at the beginning of executions only.

### 3.1 Definition

Our definition of an ideal blind signature functionality $\mathcal{F}_{\text{BlSig}}$ follows the one $\mathcal{F}_{\text{Sig}}$ of regular signature schemes given by Canetti [Can04]. The definition of $\mathcal{F}_{\text{Sig}}$ essentially lets the adversary choose the public verification key and determine the signature value $S$ upon a signing request $(\texttt{Sign}, sid, m)$. Verification requests for previously generated signatures are always accepted and otherwise the adversary is again allowed to specify whether a tested signature $S$ to a message $m$ is valid or not. See [Can04] for a discussion of this definition.

The formal description of the blind signature functionality is given in Figure 2. It is partly a verbatim copy of the functionality $\mathcal{F}_{\text{Sig}}$ in [Can04]. An important difference for blind signatures is that the adversary should not learn the message of honest users and the signatures must not be linkable to the signing request. To ensure this we let the adversary (instead of the bank, analogously to the choice of the public verification key in $\mathcal{F}_{\text{Sig}}$) in $\mathcal{F}_{\text{BlSig}}$ provide (the description of) a stateless, possibly probabilistic algorithm $\mathsf{BlSig}$ to the ideal functionality $\mathcal{F}_{\text{BlSig}}$. This is already done in the key generation step where the adversary chooses the public verification key, such that $\mathsf{BlSig}$ is used in all subsequent signature generation runs. Whenever an honest user later requests a signature for a message $m$ this algorithm $\mathsf{BlSig}(m)$ generates a signature $S$ *but without disclosing the message $m$ to the adversary*, enforcing unlinkability of signatures.

If a corrupt user —that is, the adversary on behalf of the corrupt user— requests a signature, however, the ideal functionality does not run $\mathsf{BlSig}$. Instead it asks the adversary about the potential signature $S$ this user could produce from an interaction with the bank. Note that a corrupt user may not output any signature right after an interaction with the bank (or ever), the interaction merely guarantees that this user can generate this signature in principle. Hence, the functionality does not return the adversary's potential signature $S$ to the user.

Finally, for any signature request we inform the bank $\mathcal{B}$ about the request, but without disclosing the actual message $m$ nor the signature. This captures the fact that signature generations require active participation of the bank.

**Functionality $\mathcal{F}_{\mathbf{BlSig}}$**

**Key Generation:** Upon receiving a value $(\texttt{KeyGen}, sid)$ from a party $\mathcal{B}$, verify that $sid = (\mathcal{B}, sid')$ for some $sid'$. If not, then ignore. Else, hand $(\texttt{KeyGen}, sid)$ to the adversary. Upon receiving $(\texttt{VerificationKey}, sid, pk_{\text{BS}}, \mathsf{BlSig})$ from the adversary, output $(\texttt{VerificationKey}, sid, pk_{\text{BS}})$ to $\mathcal{B}$ and record the pair $(\mathcal{B}, pk_{\text{BS}}, \mathsf{BlSig})$.

**Signature Generation:** Upon receiving a value $(\texttt{Sign}, sid, m, pk_{\text{BS}})$ for $m \in \{0,1\}^n$ from some party $\mathcal{U}$, verify that $sid = (\mathcal{B}, sid')$ for some $sid'$. If not, then ignore. Else, do the following:

- If the user $\mathcal{U}$ is honest then inform $\mathcal{B}$ and the adversary through $(\texttt{Signature}, sid)$ that a signature request takes place and then generate $S \leftarrow \mathsf{BlSig}(m)$ and output $(\texttt{Signature}, sid, m, S)$ to $\mathcal{U}$.
- If the user $\mathcal{U}$ is corrupt then send $(\texttt{Sign}, sid, m)$ to the adversary to obtain $(\texttt{Signature}, sid, m, S)$; abort if $(m, S, pk_{\text{BS}}, 0)$ has been recorded before. Send $(\texttt{Signature}, sid)$ to $\mathcal{B}$.

In either case record $(m, S, pk_{\text{BS}}, 1)$.

**Signature Verification:** Upon receiving a value $(\texttt{Verify}, sid, m, S, pk'_{\text{BS}})$ from some party $\mathcal{P}$ hand $(\texttt{Verify}, sid, m, S, pk'_{\text{BS}})$ to the adversary. Upon receiving $(\texttt{Verified}, sid, m, S, \phi)$ from the adversary do:

1. If $pk_{\text{BS}} = pk'_{\text{BS}}$ and the entry $(m, S, pk_{\text{BS}}, 1)$ is recorded, then set $f = 1$ (completeness condition).
2. Else, if $pk_{\text{BS}} = pk'_{\text{BS}}$, the bank is not corrupt, and no entry $(m, S, pk_{\text{BS}}, 1)$ is recorded, then set $f = 0$ and record the entry $(m, S, pk_{\text{BS}}, 0)$ (unforgeability condition).
3. Else, if there is an entry $(m, S, pk'_{\text{BS}}, f')$ recorded, then let $f = f'$ (consistency condition).
4. Else, let $f = \phi$ and record the entry $(m, S, pk'_{\text{BS}}, \phi)$.

Output $(\texttt{Verified}, sid, m, S, f)$ to $\mathcal{P}$.

**Figure 2.** Blind Signature Functionality $\mathcal{F}_{\text{BlSig}}$

It follows quite easily that one can realize universally composable commitment schemes in the presence of functionality $\mathcal{F}_{\text{BlSig}}$. As a consequence of the hardness of constructing such commitments [CF01,DG03] we conclude:

**Proposition 2.** *Bilateral and terminating (i.e., only two parties are active and honest parties faithfully give output) universally composable blind signature schemes securely realizing $\mathcal{F}_{BlSig}$ in the plain model do not exist. Furthermore, blind signature schemes securely realizing $\mathcal{F}_{BlSig}$ in the common reference string model imply key agreement, and imply oblivious transfer in the common random string model. This all holds for non-adaptive corruptions.*

*Proof.* Given functionality $\mathcal{F}_{\text{BlSig}}$ we build a universally composable commitment scheme in the $\mathcal{F}_{\text{BlSig}}$-hybrid model. The commitment step is as follows. The receiver calls functionality $\mathcal{F}_{\text{BlSig}}$ with $(\texttt{KeyGen}, sid)$ to generate a public key $(\texttt{VerificationKey}, sid, pk_{\text{BS}})$. It sends $pk_{\text{BS}}$ to the committer. The committer with message $m$ calls functionality $\mathcal{F}_{\text{BlSig}}$ about $(\texttt{Sign}, sid, m, pk_{\text{BS}})$ to get a signature $(\texttt{Signature}, sid, m, S)$. This completes the commitment phase. To decommit the committer sends $m, S$ to the receiver who accepts if and only if $\mathcal{F}_{\text{BlSig}}$ on $(\texttt{Verify}, sid, m, S, pk_{\text{BS}})$ returns $(\texttt{Verified}, sid, m, S, 1)$ and if it has received at most one notification $(\texttt{Signature}, sid)$.

It is not hard to see that this protocol above implements the universally composable commitment functionality $\mathcal{F}_{\text{com}}$. Hence, if there was a bilateral, terminating blind signature protocol for $\mathcal{F}_{\text{BlSig}}$ then this protocol above would realize functionality $\mathcal{F}_{\text{com}}$ according to the composition theorem. This, however, contradicts the impossibility result in [CF01]. Moreover, the result in [DG03] about the consequence of protocols realizing $\mathcal{F}_{\text{com}}$ the common *random* string model shows that such blind signatures imply key agreement. □

By the general feasibility results of Canetti et al. [Can01,CLOS02] functionality $\mathcal{F}_{\text{BlSig}}$ can be realized in the multi-party setting for honest majorities (in the plain model) and dishonest majorities (in the common random string model). Instead of relying on the general construction in [CLOS02] we construct a simpler two-move scheme in the common reference string model directly, based on the scheme in the previous section.

### 3.2 Construction

The construction in the standard model gives a good starting point for a solution in the UC framework. We augment the scheme by the following steps. First, we let the user in the first round when sending $U$ also encrypt all the data from which the proof is later derived. This ciphertext $E$ should contain the sUNIZK prover's secret $SK$ as well as randomness $u, v$ and message $m$. The encryption scheme itself needs to be IND-CPA secure and we simply use the same scheme as for the computation for $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$ but with an independent key pair $(pk'_{\mathsf{Enc}}, sk'_{\mathsf{Enc}})$.

In addition to the ciphertext $E$ the user should prove with another NIZK proof that the data is valid and matches the data committed to by $U$. For this we require *simulation-sound* NIZK proofs [Sah99,DDO+01] where a malicious prover is not able to find an accepted proof for an invalid statement, even if it sees proofs of the zero-knowledge simulator before (possibly for invalid but different theorems):

**Simulation-Soundness.** A regular NIZK proof $(\mathcal{C}, \mathcal{P}, \mathcal{V})$ for relation $R$ is called simulation-sound if for the zero-knowledge simulator $\mathcal{Z}$ and for any efficient algorithm $\mathcal{P}^*$ the following holds. Let $(crs, \sigma) \leftarrow \mathcal{Z}(\mathsf{crs}, 1^n)$ and $(x, \pi) \leftarrow \mathcal{P}^{*\mathcal{O}}(crs)$, where oracle $\mathcal{O}$ for the $i$-th call $x_i$ computes $\pi_i \leftarrow \mathcal{Z}(\mathsf{prove}, \sigma, x_i)$ and returns $\pi_i$. Then the probability that $\mathcal{V}(crs, x, \pi) = 1$ and $x \notin L_R(crs)$ and $(x, \pi) \neq (x_1, \pi_1), (x_2, \pi_2), \ldots$ is negligible (as a function of $n$).

In our case here the underlying relation $R^{\text{ss}}$ is defined by (a sequence of) circuits $\mathsf{C}_n^{\text{ss}}$ evaluating to 1 if and only if for statement $x = U||E||pk'_{\mathsf{Enc}}||crs_{\mathsf{Com}}||crs_{\text{uni}}$ it holds that $E = \mathsf{Enc}(pk'_{\mathsf{Enc}}, m||PK||u||SK||v; u')$ and $U = \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)$ and there exist a random string $\rho$ with $(PK, SK) \leftarrow \mathcal{P}_{\text{uni}}(\mathsf{key}, crs_{\text{uni}}; \rho)$. Note that the length of the statement as well as the size of the witness are bounded by some fixed polynomial in $n$, because the length of the data for the sUNIZK proof can be described by a some polynomial in $n$, $c(n)$, $e(n)$, $h(n)$ and $s(n)$. Such simulation-sound NIZK proofs exist if trapdoor permutations exist [DDO+01].

The final step is to make the signature algorithm $\mathsf{Sig}$ of the bank's strongly unforgeable signature scheme deterministic. This can be accomplished by adding a key of a pseudorandom function to the secret signing key. Each time a signature for $U$ is requested the signing algorithm first applies the pseudorandom function to $U$ to get the randomness $s$ with which the signature $B \leftarrow \mathsf{Sig}(sk_{\mathsf{Sig}}, U; s)$ is computed deterministically. The advantage of this modification, which does not require an additional complexity assumption, is that identical requests are now also answered identically. For the same consistency reason we also presume that the verification algorithm of the sUNIZK proof is deterministic (as for example in our construction).

| **Bank $\mathcal{B}$** | $crs_{\text{BS}}$, $pk_{\text{BS}}$ | **User $\mathcal{U}$** |
|---|---|---|
| signing key $sk_{\mathsf{Sig}}$ | | message $m \in \{0,1\}^n$ |
| | | choose $u, v, u' \leftarrow \{0,1\}^n$<br>let $(PK, SK) \leftarrow \mathcal{P}_{\text{uni}}(\mathsf{key}, crs_{\text{uni}}; \rho)$<br>let $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)$<br>let $E \leftarrow \mathsf{Enc}(pk'_{\mathsf{Enc}}, m||PK||v||SK||u; u')$<br>let $x_{\text{ss}} \leftarrow U||E||pk'_{\mathsf{Enc}}||crs_{\mathsf{Com}}||crs_{\text{uni}}$<br>let $w_{\text{ss}} \leftarrow m||PK||u||SK||v||u'||\rho$<br>compute $\pi_{\text{ss}} \leftarrow \mathcal{P}_{\text{ss}}(crs_{\text{ss}}, x_{\text{ss}}, w_{\text{ss}})$ |
| | $\xleftarrow{U, E, \pi_{\text{ss}}}$ | |
| $x_{\text{ss}} \leftarrow U||E||pk'_{\mathsf{Enc}}||crs_{\mathsf{Com}}||crs_{\text{uni}}$<br>abort if $\mathcal{V}_{\text{ss}}(crs_{\text{ss}}, x_{\text{ss}}, \pi_{\text{ss}}) \neq 1$<br>let $B \leftarrow \mathsf{Sig}(sk, U)$ | | |
| | $\xrightarrow{B}$ | abort if $\mathsf{Vf}_{\mathsf{Sig}}(pk_{\mathsf{Sig}}, U, B) \neq 1$<br>compute $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$<br>set $x \leftarrow C||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK)||m$<br>set $w \leftarrow u||v||B$<br>let $\pi \leftarrow \mathcal{P}_{\text{uni}}(\mathsf{prove}, crs_{\text{uni}}, x, w, SK)$<br>set $S \leftarrow C||PK||\pi$ |

**Figure 3.** UC Blind Signature Scheme: Issue Protocol

We note that our protocol is defined in the common reference string model. In contrast to the case of UC commitments in [CF01] where a fresh common reference string for each

commitment through $\mathcal{F}_{\text{Com}}$ is required, in our case the once generated common reference string can be used for *several* signature generations by different users; we merely need an independent common reference string for each party taking the role of a bank.

**Construction 3 (Universally Composable Blind Signatures).** *Let* $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$ *be a signature scheme,* $(\mathsf{KG}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme,* $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ *be a commitment scheme, and* $\mathcal{H}$ *be a hash function family. Let* $(\mathcal{C}_{uni}, \mathcal{P}_{uni}, \mathcal{V}_{uni})$ *be a non-interactive zero-knowledge proof system for* $R^{BS}$ *and let* $(\mathcal{C}_{ss}, \mathcal{P}_{ss}, \mathcal{V}_{ss})$ *be a non-interactive zero-knowledge proof system for* $R^{ss}$*. Define the following four procedures:*

**CRS Generation.** *Algorithm* $\mathcal{C}_{BS}$ *on input* $1^n$ *runs* $crs_{uni} \leftarrow \mathcal{C}_{uni}(1^n)$*,* $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$*,* $crs_{ss} \leftarrow \mathcal{C}_{ss}(1^n)$ *and pairs* $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}), (pk'_{\mathsf{Enc}}, sk'_{\mathsf{Enc}}) \leftarrow \mathsf{KG}_{\mathsf{Enc}}(1^n)$*. It outputs* $crs_{BS} \leftarrow (crs_{uni}, crs_{\mathsf{Com}}, pk_{\mathsf{Enc}}, pk'_{\mathsf{Enc}}, crs_{ss})$*.*

**Key Generation.** *If party* $\mathcal{B}$ *with access to* $crs_{BS}$ *receives the input* $(\texttt{KeyGen}, sid)$ *it checks that* $sid = (\mathcal{B}, sid')$ *for some* $sid'$*. If not, it ignores. Else it generates a signature key pair* $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) \leftarrow \mathsf{KG}_{\mathsf{Sig}}(1^n)$ *and picks a hash function* $H \leftarrow \mathcal{H}(1^n)$*. It sets* $(pk_{BS}, sk_{BS}) \leftarrow ((pk_{\mathsf{Sig}}, H), sk_{\mathsf{Sig}})$*, stores* $sk_{BS}$*and outputs* $(\texttt{VerificationKey}, sid, pk_{\text{BS}})$*.*

**Signature Issue Protocol.** *If party* $\mathcal{U}$ *is invoked with input* $(\texttt{Sign}, sid, m, pk_{\text{BS}})$ *for* $sid = (\mathcal{B}, sid')$ *it initiates a run of the interactive protocol in Figure 3 with the bank* $\mathcal{B}$*, where the user gets* $m$ *and* $pk_{BS}$ *as input and the bank uses* $sk_{BS}$ *as input. The user outputs* $(\texttt{Signature}, sid, m, S)$ *for the derived signature value* $S$*.*

**Signature Verification.** *If a party receives* $(\texttt{Verify}, sid, m, S, pk'_{\text{BS}})$ *it parses* $S$ *as* $S = C||PK||\pi$*, computes* $\phi \leftarrow \mathcal{V}(crs, PK, x, \pi)$ *for* $x = C||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK)||m$ *and outputs* $(\texttt{Verified}, sid, m, \sigma, \phi)$*.*

**Theorem 4.** *Let* $(\mathsf{KG}_{\mathsf{Sig}}, \mathsf{Sig}, \mathsf{Vf}_{\mathsf{Sig}})$ *be a length-invariant signature scheme which is strongly unforgeable against adaptive chosen-message attacks and for which* $\mathsf{Sig}$ *is deterministic. Let* $(\mathsf{KG}_{\mathsf{Enc}}, \mathsf{Enc}, \mathsf{Dec})$ *be a length-invariant IND-CPA secure encryption scheme,* $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ *be a length-invariant non-interactive commitment scheme with unique openings in the common reference string model. Also let* $\mathcal{H}$ *be a collision-intractable hash function family and* $(\mathcal{C}_{uni}, \mathcal{P}_{uni}, \mathcal{V}_{uni})$ *be a single-theorem unique non-interactive zero-knowledge proof system for* $R^{BS}$ *with deterministic verifier* $\mathcal{V}_{uni}$*. Let* $(\mathcal{C}_{ss}, \mathcal{P}_{ss}, \mathcal{V}_{ss})$ *be a (regular) simulation-sound non-interactive zero-knowledge proof for* $R^{ss}$*. Then the scheme defined in Construction 3 securely realizes functionality* $\mathcal{F}_{BlSig}$ *for non-adaptive corruptions.*

The idea of the proof is as follows. Algorithm $\mathsf{BlSig}$ for functionality $\mathcal{F}_{\text{BlSig}}$, supposed to be determined by the ideal-model adversary and to create signatures for honest users, ignores its input $m$ entirely, but instead prepares a dummy encryption $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, 0^{c(n)+s(n)}; v)$ and appends a fake correctness proof $PK||\pi$ generated by the zero-knowledge simulator. The output $C||PK||\pi$ is thus indistinguishable from genuinely generated signatures of honest users in the actual scheme. On the other hand, the additional encryption $E$ and the simulation-sound zero-knowledge proof allows the ideal-model adversary to extract potential signatures $C||PK||\pi$ of malicious users (in black-box simulations), and to provide them to the functionality. The completeness and consistency condition of $\mathcal{F}_{\text{BlSig}}$ is realized by the completeness of the underlying scheme, and unforgeability follows as for the basic scheme with concurrent security.

*Proof.* (of Theorem 4) We have to show that for each adversary $\mathcal{A}$ attacking the real-world protocol there exist an ideal-model adversary (aka. simulator) $\mathcal{S}$ in the ideal world with

dummy parties and functionality $\mathcal{F}_{\text{BlSig}}$ such that no environment $\mathcal{Z}$ can distinguish whether it is facing an execution in the real world with $\mathcal{A}$ or one in the ideal world with $\mathcal{S}$.

We build the ideal-model adversary $\mathcal{S}$ by black-box simulation of $\mathcal{A}$, relaying all communication between the environment $\mathcal{Z}$ and the (simulated) adversary $\mathcal{A}$, and acting on behalf of the honest parties in this simulation. Algorithm $\mathcal{S}$ also corrupts a dummy party in the ideal model whenever $\mathcal{A}$ asks to corrupt the corresponding party in the simulation. By assumption this is only done before the execution starts.

The ideal-model simulator $\mathcal{S}$ first generates a reference string $crs_{\text{BS}}$ for the black-box simulation by picking encryption keys $(sk_{\mathsf{Enc}}, pk_{\mathsf{Enc}}), (sk'_{\mathsf{Enc}}, pk'_{\mathsf{Enc}}) \leftarrow \mathsf{KG}_{\mathsf{Enc}}(1^n)$, generating $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$ and running the zero-knowledge simulators to generates $crs_{\text{ss}}$ as well as $crs_{\text{uni}}$ for the unique zero-knowledge proof. It outputs $crs_{\text{BS}} = (pk_{\mathsf{Enc}}, pk'_{\mathsf{Enc}}, crs_{\text{ss}}, crs_{\mathsf{Com}}, crs_{\text{uni}})$. We next describe the simulation of the honest parties in the black-box simulation:

- Suppose the simulator is woken up through a call $(\texttt{KeyGen}, sid)$ from $\mathcal{F}_{\text{BlSig}}$ in the ideal model where $sid = (\mathcal{B}, sid')$. Then the simulator generates $(pk_{\text{BS}}, sk_{\text{BS}}) \leftarrow \mathsf{KG}_{\text{BS}}(1^n)$ as specified by the scheme's description and lets $\mathsf{BlSig}$ be the algorithm that on input $m \in \{0,1\}^n$ computes the encryption $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, 0^{c(n)+s(n)}; v)$, $(PK, SK) \leftarrow \mathcal{Z}(\mathsf{key}, \sigma)$ and $\pi$ through the zero-knowledge simulator for relation $R^{\text{BS}}$ where the statement $x$ is defined by $x \leftarrow C||pk_{\mathsf{Enc}}||crs_{\mathsf{Com}}||pk_{\mathsf{Sig}}||H(PK)||m$, and finally outputs $S \leftarrow C||PK||\pi$. Simulator $\mathcal{S}$ returns $(\texttt{VerificationKey}, sid, pk_{\text{BS}}, \mathsf{BlSig})$ to $\mathcal{F}_{\text{BlSig}}$ in the ideal model. In the black-box simulation it sends $pk_{\text{BS}}$ to all parties.
- Suppose that the adversary lets a corrupt user in the black-box simulation initiate a protocol run with the honest bank by sending values $(U, E, \pi_{\text{ss}})$. Then the simulator first checks the validity of the proof $\pi_{\text{ss}}$; if this check fails then it ignores this message. Else, $\mathcal{S}$ uses the secret key $sk'_{\mathsf{Enc}}$ to recover $m||PK||v||SK||u$ from $E$ (and aborts if it fails) and submits $(\texttt{Sign}, sid, m, pk_{\text{BS}})$ on behalf of the user to the ideal functionality. It immediately receives a request $(\texttt{Signature}, sid, m)$ from $\mathcal{F}_{\text{BlSig}}$. To answer, $\mathcal{S}$ computes the signature $B \leftarrow \mathsf{Sig}(sk_{\mathsf{Sig}}, U)$ under the strongly unforgeable signature scheme, an encryption $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$ and a proof $\pi \leftarrow \mathcal{P}(crs_{\text{uni}}, x, w, SK)$ for the extracted values and sends $(\texttt{Signature}, sid, m, S)$ for $S \leftarrow C||PK||\pi$ back to the functionality. It also returns $B$ in the black-box simulation to the corrupt user.
- If an honest user requests a signature $(\texttt{Sign}, sid, m, pk_{\text{BS}})$ in the ideal model and waits to receive $(\texttt{Signature}, sid, m, S)$, generated by the functionality through algorithm $\mathsf{BlSig}$, then the ideal-model adversary generates strings $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, 0^{2n+h(n)}; u)$, $E \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, 0 \ldots 0; u')$ and a proof $\pi_{\text{ss}}$ via the zero-knowledge simulator of the simulation-sound scheme and lets the user in the black-box simulation send these values $(U, E, \pi_{\text{ss}})$. If the bank is honest, then $\mathcal{S}$ uses the secret signing key $sk_{\mathsf{Sig}}$ to compute $B \leftarrow \mathsf{Sig}(sk_{\mathsf{Sig}}, U)$, else it waits to receive a value $B$ from the adversarial controlled bank.
- If $\mathcal{S}$ in the ideal model gets a request $(\texttt{Verify}, sid, m, S, pk'_{\text{BS}})$ then it computes $\phi \leftarrow \mathcal{V}(crs, PK, x, \pi)$ and returns $(\texttt{Verified}, sid, m, S, \phi)$.

This gives a full description of the ideal-model simulator. For the analysis note that there are two main differences between the ideal-model and the black-box simulation compared to an actual attack on the protocol. First, in the black-box simulation we use fake values (commitments and encryptions of zero-strings, simulated proofs etc.). The second point to address is that the verification algorithm in the ideal model returns 0 if there is no recorded pair $(m, S, pk_{\text{BS}}, 1)$ while in the real-life protocol $\mathsf{Vf}_{\text{BS}}$ may output 1; for any other verification requests the answers are identical as the verification algorithm $\mathsf{Vf}_{\text{BS}}$ merely runs the deter-

ministic verification algorithm of the unique NIZK system (and thus guarantees completeness and, especially, consistency).

We claim that the differences are undetectable for the environment $\mathcal{Z}$. This is proven through a sequence of games transforming an execution in the ideal-model scenario into one which is equal to the one of the actual protocol. In these games we will have full control over the setting, in particular over the functionality and, in contrast to the ideal-model adversary, we will also read the inputs of $\mathcal{Z}$ to honest users. This is admissible since our goal is to emulate $\mathcal{Z}$'s environment and to use differences in the output behavior to contradict the security of the underlying cryptographic primitives.

- Experiment $\mathsf{Game}_0(n)$ describes the original attack of $\mathcal{Z}$ on the ideal-model simulation (including the black-box simulation of $\mathcal{A}$).
- In $\mathsf{Game}_1(n)$ we change the way the commitments $U$ on behalf of honest users are computed in $\mathcal{A}$'s black-box simulation. Originally, the simulator $\mathcal{S}$ computes a fake commitment $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, 0^{2n+h(n)}; u)$. Now, whenever the simulator is supposed to create such a commitment, we let $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)$ for the right value $m$ (given as input to the honest user by $\mathcal{Z}$), $(PK, SK)$ generated by the zero-knowledge simulator for the sUNIZK proof and $u, v \leftarrow \{0,1\}^n$ picked at random. Because of the secrecy of $\mathsf{Com}$ it is easy to see that $\mathcal{Z}$'s output behavior will not change significantly when facing $\mathsf{Game}_1(n)$ instead of $\mathsf{Game}_0(n)$.
- Next, in $\mathsf{Game}_2(n)$, we replace every encryption $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, 0^{c(n)+s(n)}; v)$ in the computations of algorithm $\mathsf{BlSig}$ through an encryption of the actual values $U||B$, i.e., $C \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, U||B; v)$, where we provide the values $U$ and $B$ transmitted in the black-box simulation "adaptively" to algorithm $\mathsf{BlSig}$. By the security of the encryption scheme this is indistinguishable from the environment's viewpoint.
- In $\mathsf{Game}_3(n)$ we replace every steps of the zero-knowledge proof in the computation of $\mathsf{BlSig}$ through steps of the actual proof system, i.e., generation of $crs_{\mathrm{uni}}$ through $crs_{\mathrm{uni}} \leftarrow \mathcal{C}_{\mathrm{uni}}(1^n)$, and every generation of $PK$ and $\pi$ through the prover (for the now genuine witness $w = u||v||B$). By the zero-knowledge property this substitution is (computationally) indistinguishable for the environment $\mathcal{Z}$.
- Now we turn to the difference between the ideal-model verification through list comparisons and the real-life verification through $\mathsf{Vf}_{\mathrm{BS}}$. In $\mathsf{Game}_4(n)$ every time the verification algorithm in $\mathsf{Game}_3(n)$ was called by some honest user about input $(\texttt{Verify}, sid, m, S, pk_{\mathrm{BS}})$ then we run the verification $\mathsf{Vf}_{\mathrm{BS}}(crs_{\mathrm{BS}}, pk_{\mathrm{BS}}, m, S)$ instead (the case $pk'_{\mathrm{BS}} \neq pk_{\mathrm{BS}}$ is treated as before).
  Consider in $\mathsf{Game}_3(n)$ the event that some user requests the functionality to verify a signature $(\texttt{Verify}, sid, m, S, pk_{\mathrm{BS}})$ such that there is no entry $(m, S, pk_{\mathrm{BS}}, 1)$ stored by the functionality (and the bank $\mathcal{B}$ is honest) but such that $\mathsf{Vf}_{\mathrm{BS}}$ returns 1. This would clearly allow to distinguish the two games (note that the other direction, that $\mathsf{Vf}_{\mathrm{BS}}$ yields 0 but there is an entry, cannot happen by the completeness of the blind signature scheme).
  We claim that if such a request should occur with noticeable probability this would contradict the unforgeability of the blind signature scheme in the previous section (there, unforgeability was proven for arbitrary, strongly unforgeable signature scheme for the bank, and thus holds for the deterministic one we consider here as well).
  Specifically, we show how to turn a run of $\mathsf{Game}_3(n)$ with $\mathcal{A}$ and $\mathcal{Z}$ into an attack according to experiment $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}(n)$. For this we run the same experiment as in $\mathsf{Game}_3(n)$ but this time we are use the oracle access to the bank's signing oracle $\mathsf{Sig}(sk_{\mathsf{Sig}}, \cdot)$ instead of generating the key pair $(sk_{\mathsf{Sig}}, pk_{\mathsf{Sig}})$ ourselves (recall that the bank is assumed to be honest). Each time an honest user receives input $(\texttt{Sign}, sid, m, pk_{\mathrm{BS}})$ we generate $U, E, \pi_{\mathrm{ss}}$

as in $\mathsf{Game}_3(n)$, including a valid commitment $U \leftarrow \mathsf{Com}(crs_{\mathsf{Com}}, m||H(PK)||v; u)$, and submit $U$ to the bank to receive $B$. From this answer we honestly compute the signature $S \leftarrow C||PK||\pi$ with the help of $SK$. We memorize the pair $(m, S)$ and the values $(U, B)$.
If a corrupt user submits $U, E, \pi_{\text{ss}}$ in the black-box simulation then we also check the proof $\pi_{\text{ss}}$ (and do nothing if it is invalid). If the proof is accepted then we check if we have stored a pair $(U, B)$ for some $B$. If so, we return the same $B$ as before in the black-box simulation but without contacting the bank in the attack $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}(n)$ (since the bank's signature would be identical we answer consistently). If there is no such pair $(U, B)$ we use $sk'_{\mathsf{Enc}}$ to extract appropriate values $m||PK||v||SK||u$. By the simulation soundness of the NIZK this extraction works with overwhelming probability (because $U$ has never appeared for an honest user in the execution before). We submit $U$ to the bank's signature oracle to receive a value $B$, which we return to the corrupt user in the black-box simulation. We also deduce the signature $S$ with the help of the extracted values and record $(m, S)$ and $(U, B)$.
Suppose now that a user at some point sends a request $(\texttt{Verify}, sid, m, S, pk_{\text{BS}})$ for a pair $(m, S)$ which we have not stored but for which $\mathsf{Vf}_{\text{BS}}$ accepts. Then we immediately stop and output all previously stored $k$ message/signature pairs together with $(m, S)$. Note that this implies that all our $k + 1$ pairs are accepted by $\mathsf{Vf}_{\text{BS}}$, although we only had $k$ interactions with the bank. Hence, if a mismatch in $\mathsf{Game}_3(n)$ happens with noticeable probability it would refute the unforgeability of the blind signature scheme of the previous section. It follows that $\mathsf{Game}_3(n)$ and $\mathsf{Game}_4(n)$ are indistinguishable.

– In $\mathsf{Game}_5(n)$ we can omit the extraction step where the ideal-model simulator decrypts $E$ from submissions of corrupt users, in particular, we do not need to know the secret key $sk'_{\mathsf{Enc}}$ anymore for the simulation. This is so since the verification now only relies on $\mathsf{Vf}_{\text{BS}}$ instead of lists. Furthermore every time we gave a dummy encryption $E \leftarrow \mathsf{Enc}(pk'_{\mathsf{Enc}}, 0 \ldots 0; u')$ for an honest user, we now encrypt the true values $E \leftarrow \mathsf{Enc}(pk'_{\mathsf{Enc}}, m||PK||v||SK||u; u')$ to prepare the correct values $U$ and $C$. By the security of the encryption scheme this is indistinguishable for the environment.
– In $\mathsf{Game}_6(n)$ we replace the simulation of proofs $\pi_{\text{ss}}$ for honest users through proofs computed by the prover's algorithm for witness $m||PK||v||SK||u$ (with respect to a truly random string $crs_{\text{ss}}$). The zero-knowledge property ensures that this will not significantly affect the environment's output.

All the steps in the final game now are exactly as in an attack on the real protocol with adversary $\mathcal{A}$. Therefore, the environment's output in the ideal-model simulation ($\mathsf{Game}_0(n)$) and the real-world execution ($\mathsf{Game}_6(n)$) are indistinguishable. □

Similarly to the case of the underlying blind signature scheme we can weaken the security requirement on the unforgeability and allow a malicious user to be able to generate multiple blind signature for a message. In this case we first have to change the the verification step 2 in the definition of $\mathcal{F}_{\text{BlSig}}$ as follows:

2. Else, if $pk_{\text{BS}} = pk'_{\text{BS}}$, the bank is not corrupt, and no entry $(m, \star, pk_{\text{BS}}, 1)$ is recorded, then set $f = 0$ and record the entry $(m, S, pk_{\text{BS}}, 0)$ (unforgeability condition).

As in the case of our concurrently secure scheme we can then assume that the bank's signing algorithm is unforgeable in the weaker sense (but still deterministic) and drop the requirement on the uniqueness of the NIZK proof and remove the hash function. The corresponding scheme then realizes this weaker notion of $\mathcal{F}_{\text{BlSig}}$.

Moreover, we can also extend the functionality $\mathcal{F}_{\text{BlSig}}$ to handle partially blind signatures. For this we let the user provide some public information info together with the message $m$ for signature requests $(\texttt{Sign}, sid, m, pk_{\text{BS}})$. This information is then forwarded to the adversary and to the bank by the functionality, and we include this information info as part of the maintained records. We can realize this functionality by starting with the partially blind version of our concurrently-secure blind signature scheme and adding the encryption and the simulation-sound proof as before.

## Acknowledgment

We thank the anonymous reviewers for comprehensive comments.

## References

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

[AF96] Masayuki Abe and Eiichiro Fujisaki. *How to Date Blind Signatures*. Advances in Cryptology — Asiacrypt'96, Volume 1163 of Lecture Notes in Computer Science, pages 244–251. Springer-Verlag, 1996.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. *Noninteractive Zero-Knowledge. SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. *The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. Journal of Cryptology*, 16(3):185–215, 2003.

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

[Can01] Ran Canetti. *Universally Composable Security: A new Paradigm for Cryptographic Protocols*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 2001. IEEE Computer Society Press, for an updated version see `eprint.iacr.org`, 2001.

[Can04] Ran Canetti. *On Universally Composable Notions of Security for Signature, Certification and Authentication*. Proceedings of Computer Security Foundations Workshop (CSFW) 2004. IEEE Computer Society Press, for an updated version see `eprint.iacr.org`, 2004.

[CF01] Ran Canetti and Marc Fischlin. *Universally Composable Commitments*. Advances in Cryptology — Crypto 2001, Volume 2139 of Lecture Notes in Computer Science, pages 19–40. Springer-Verlag, 2001.

[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.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. *Universally Composable Two-Party and Multi-Party Secure Computation*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2002, pages 494–503. ACM Press, 2002.

[Dam93] Ivan Damgård. *Non-Interactive Circuit Based Proofs and Non-Interactive Perfect Zero-knowledge with Proprocessing*. Advances in Cryptology — Eurocrypt'92, Volume 658 of Lecture Notes in Computer Science, pages 341–355. Springer-Verlag, 1993.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. *Robust Non-interactive Zero Knowledge*. Advances in Cryptology — Crypto 2001, Volume 2139 of Lecture Notes in Computer Science, pages 566–598. Springer-Verlag, 2001.

[DG03] Ivan Damgård and Jens Groth. *Non-interactive and Reusable Non-Malleable Commitment Schemes*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2003, pages 426–437. ACM Press, 2003.

[DIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. *Non-interactive and Non-Malleable Commitment*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1998, pages 141–150. ACM Press, 1998.

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. *Non-Interactive Zero-Knowledge with Preprocessing*. Advances in Cryptology — Crypto'88, Volume 403 of Lecture Notes in Computer Science, pages 269–282. Springer-Verlag, 1988.

[DP92] Alfredo De Santis and Giuseppe Persiano. *Zero-Knowledge Proofs of Knowledge Without Interaction*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)'92, pages 427–436. IEEE Computer Society Press, 1992.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. *Multiple NonInteractive Zero Knowledge Proofs Under General Assumption*. *SIAM Journal on Computing*, 29(1):1–28, 1999.

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

[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.

[Lin03] Yehuda Lindell. *Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2003, pages 683–692. ACM Press, 2003.

[Lin04] Yehuda Lindell. *Lower Bounds for Concurrent Self Composition*. Theory of Cryptography Conference (TCC) 2004, Volume 2951 of Lecture Notes in Computer Science, pages 203–222. Springer-Verlag, 2004.

[LMS05] Matt Lepinski, Silvio Micali, and Abhi Shelat. *Fair Zero-Knowledge*. Theory of Cryptography Conference (TCC) 2005, Volume 3378 of Lecture Notes in Computer Science, pages 245–263. Springer-Verlag, 2005.

[NY89] Moni Naor and Moti Yung. *Universal One-Way Hash Functions and Their Cryptographic Applications*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1989, pages 33–43. ACM Press, 1989.

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

[Poi98] David Pointcheval. *Strengthened Security for Blind Signatures*. Advances in Cryptology — Eurocrypt'98, Volume 1403 of Lecture Notes in Computer Science, pages 391–405. Springer-Verlag, 1998.

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

[Rom90] John Rompel. *One-Way Functions are Necessary and Sufficient for Secure Signatures*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1999, pages 387–394. ACM Press, 1990.

[Sah99] Amit Sahai. *Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security*. Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 1999. IEEE Computer Society Press, 1999.

# A Single-Theorem Unique NIZK Proofs for CircuitSAT

In this section we first recall the definition of unique zero-knowledge from [LMS05] and adapt it to the single-theorem case. We then present our construction of single-theorem unique zero-knowledge proofs. For this we first give the precise definition of such proofs, then describe the construction idea and tools, and finally present the construction and its security.

## A.1 Definition

In the definition below we follow the classical approach of non-interactive zero-knowledge proofs where a public string *crs* is available to all parties. Here we use the more general definition of an algorithm $\mathcal{C}(1^n)$ generating this string *crs*, which is either called the common random string (if *crs* is a uniformly distributed bit string of polynomial length in $n$) or the common reference string (if the string has an arbitrary distribution).

Single-theorem UNIZK proofs obey first of all the two basic properties completeness (the verifier $\mathcal{V}$ accepts all honestly generated proofs of the prover $\mathcal{P}$) and soundness (no malicious prover can make the verifier accepts proofs for invalid statements). In addition, the proof should be multiple zero-knowledge, i.e., the following two cases are indistinguishable: In the first case each invocation of the prover starts $\mathcal{P}$ in mode $\mathsf{key}$ to generate a public key $PK_i$, together with a secret key $SK_i$, and then $\mathcal{P}$ learns the valid theorem $x_i$ and the witness $w_i$ and generates the proof $\pi_i$ from $x_i, w_i$ and $SK_i$ in mode $\mathsf{prove}$. In the other case a simulator $\mathcal{Z}$ (which is also allowed to generated *crs* instead of $\mathcal{C}$) like the prover first generates $PK_i$ and $SK_i$ in mode $\mathsf{key}$ but then only gets to see $x_i$ (but not the witness) and is supposed to output a proof $\pi_i$ in mode $\mathsf{prove}$. In both cases the parties can be invoked an unbounded, but polynomially number of times.

Uniqueness now says that for each $PK$, even if maliciously chosen, for any $x$ there exist one and only one proof $\pi$ for each witness $w$. This should hold with overwhelming probability over the choice of $crs \leftarrow \mathcal{C}(1^n)$. Following [LMS05] we define uniqueness by a bijection between witnesses and valid proofs (if any).

For the sUNIZK proofs we parameterize the underlying $\mathcal{NP}$ relation $R$ by the common reference string *crs* and the public key $PK$. These parameters are both themselves determined according to a complexity parameter $n$, and for such parameters $R_{crs,PK}$ takes as inputs $x \in \{0,1\}^{\chi(n)}$ and $w \in \{0,1\}^{\omega(n)}$, where $x, w$ may depend on $crs, PK$. Let $W_{crs,PK}(x) = \left\{w \in \{0,1\}^{\omega(n)} \;\middle|\; R_{crs,PK}(x,w) = 1\right\}$ denote the set of witnesses to $x$ with respect to *crs* and $PK$, and by $L_R(crs, PK) = \left\{x \in \{0,1\}^{\chi(n)} \;\middle|\; W_{crs,PK}(x) \neq \emptyset\right\}$ the inputs in the language (for parameters $crs, PK$).

**Definition 4 (Single-Theorem Unique Zero-Knowledge).** *A single-theorem unique non-interactive zero-knowledge (sUNIZK) proof system for an efficient relation $R$ in the common reference string model is a tuple $(\mathcal{C}, \mathcal{P}, \mathcal{V})$ of efficient algorithms such that the following holds.*

**Completeness.** *For any parameter $n \in \mathbb{N}$, any $crs \leftarrow \mathcal{C}(1^n)$, any $(PK, SK) \leftarrow \mathcal{P}(\mathsf{key}, crs)$, any $(x,w) \in R_{crs,PK}$ and any $\pi \leftarrow \mathcal{P}(\mathsf{prove}, crs, x, w, SK)$ we have $\mathcal{V}(crs, x, PK, \pi) = 1$.*

**Soundness.** *For any efficient algorithm $\mathcal{P}^*$ the following holds. Let $crs \leftarrow \mathcal{C}(1^n)$ and let $(PK, x, \pi) \leftarrow \mathcal{P}^*(crs)$. Then the probability of $\mathcal{V}(crs, x, PK, \pi) = 1$ despite $x \notin L_R(crs, PK)$ is negligible (as a function of $n$).*

**Zero-Knowledge.** *There exists a probabilistic polynomial-time algorithm $\mathcal{Z}$ such that for all probabilistic polynomial-time algorithms $\mathcal{D}$ the following random variable (as functions of $n$) are computationally indistinguishable:*

- *Let* $crs \leftarrow \mathcal{C}(1^n)$, $(PK_1, SK_1) \leftarrow \mathcal{P}(\mathsf{key}, crs)$ *and output* $d \leftarrow \mathcal{D}^{\mathcal{O}}(crs, PK_1)$, *where (stateful) oracle* $\mathcal{O}$ *for the* $i$*-th call* $(x_i, w_i)$ *first computes a genuine proof* $\pi_i \leftarrow \mathcal{P}(\mathsf{prove}, crs, x_i, w_i, SK_i)$ *if* $(x_i, w_i) \in R_{crs,PK_i}$ *and sets* $\pi_i \leftarrow \perp$ *else, and also computes* $(PK_{i+1}, SK_{i+1}) \leftarrow \mathcal{P}(\mathsf{key}, crs)$, *and the oracle returns* $(\pi_i, PK_{i+1})$.
- *Let* $(crs, \sigma) \leftarrow \mathcal{Z}(\mathsf{crs}, 1^n)$, $(PK_1, SK_1) \leftarrow \mathcal{Z}(\mathsf{key}, \sigma)$ *and output* $d \leftarrow \mathcal{D}^{\mathcal{O}}(crs, PK_1)$, *where (stateful) oracle* $\mathcal{O}$ *for the* $i$*-th call* $(x_i, w_i)$ *computes a simulated proof* $\pi_i \leftarrow \mathcal{Z}(\mathsf{prove}, \sigma, x_i, w_i, SK_i)$ *if* $(x_i, w_i) \in R_{crs,PK}$ *and sets* $\pi_i \leftarrow \perp$ *else, and also computes* $(PK_{i+1}, SK_{i+1}) \leftarrow \mathcal{Z}(\mathsf{key}, \sigma)$, *and the oracle returns* $(\pi_i, PK_{i+1})$.

**Uniqueness.** *With overwhelming probability over the choice of* $crs \leftarrow \mathcal{C}(1^n)$, *for any* $PK$ *and any* $x$, *if the set* $\Pi_{crs,PK}(x) = \{\pi \mid \mathcal{V}(crs, x, PK, \pi) = 1\}$ *of accepted proofs in not empty, then there exists a bijection* $\tau_{crs,PK,x}$ *between the set* $\Pi_{crs,PK}(x)$ *and the set* $W_{crs,PK}(x) = \{w \mid (x, w) \in R_{crs,PK}\}$ *of witnesses.*

*If, in addition,* $\mathcal{C}(1^n)$ *generates uniformly distributed bit strings then the proof is said to be in the common random string model. If the proof system is for an* $\mathcal{NP}$*-complete relation* $R$ *then we say that the proof system is for* $\mathcal{NP}$. *If we drop the requirement for uniqueness then we simply speak of a (regular) non-interactive zero-knowledge proof system.*

Below we present a scheme which is zero-knowledge for a single proof, i.e., where the zero-knowledge property holds with respect to distinguisher $\mathcal{D}$ querying oracle $\mathcal{O}$ only once. By a well-known technique due to Feige et al. [FLS99] we can turn such proof into a multiple zero-knowledge system assuming the existence of one-way functions. It is not hard to see that this transformation preserves uniqueness (as we will discuss briefly at the end). Since the technique already applies if the starting protocol is witness-indistinguishable instead of zero-knowledge, i.e.,

**Witness Indistinguishability.** For any efficient algorithm $\mathcal{D}$ the following holds. Let $crs \leftarrow \mathcal{C}(1^n)$, $(PK, SK) \leftarrow \mathcal{P}(\mathsf{key}, crs)$, $(x, w_0, w_1, \delta) \leftarrow \mathcal{D}(crs, PK)$ such that $w_0, w_1 \in W_{crs,PK}(x)$, $b \leftarrow \{0, 1\}$, $\pi \leftarrow \mathcal{P}(\mathsf{prove}, crs, x, w_b, SK)$, and $d \leftarrow \mathcal{D}(\delta, \pi)$. Then the probability that $d = b$ is negligibly close to $1/2$.

it suffices to prove our protocol to be a single-theorem unique witness-indistinguishable non-interactive proof system.

### A.2 Construction Idea

Our solution is influenced by the construction in [BDMP91] and [LMS05], yet relies on two important differences:

The first difference is to use Damgård's construction [Dam93] based on CircuitSAT instead of relying on the 3SAT protocol of [BDMP91,LMS05]. The CircuitSAT problem is to decide whether a circuit $\mathtt{C}$ is satisfiable or not. Switching to this problem gains us an important advantage, based on the $\mathcal{NP}$ hardness proof of this problem. Let $M$ be a Turing machine of an $\mathcal{NP}$ language $L$ which on input $(x, w)$ verifies that $w$ is a valid witness for $x$. Then the Karp reduction from $L_R$ to CircuitSAT transforms $M$ into a circuit $\mathtt{C}_x$ with $x$ hardwired into the circuit's description. The circuit $\mathtt{C}_x$ itself is basically a description of $M$'s program, mapping inputs $w$ to $M$'s decision $M(x, w) \in \{0, 1\}$. Instead of fixing $x$ with $\mathtt{C}_x$ we now contemplate $x$ to be part of the input, and consider circuit $\mathtt{C}_n$ (for some bounded input length $n$) which takes $(x, w)$ as input and computes $M(x, w)$. In particular, $\mathtt{C}_n$ now depends only on the language $L_R$ but not the actual statement. This enables us to build a "scrambled" circuit computation

as in [Dam93] *before $x$ is actually known*. A similar idea has been applied in the case of 3SAT in [DMP88].

The second important difference, allowing us to base our protocol on general assumptions instead of quadratic residuosity as in [BDMP91,Dam93,LMS05], is to use an equivocal commitment scheme under general assumptions. Such an equivocal commitment scheme is binding for any (possibly malicious) prover, but allows a simulator to open the commitment in any arbitrary way. This idea is implicit in [BDMP91,Dam93,LMS05] where the prover is given a quadratic non-residue $y$ and commits to a bit $b$ by $r^2y^b \bmod N$ for random $r$. If the zero-knowledge simulator chooses $y$ as a square instead then such a commitment can be opened correctly for any value $b \in \{0,1\}$.

The same functionality as in the quadratic residuosity construction can be achieved with the equivocal commitment scheme of Di Crescenzo et al. [DIO98]. There, a random string $t$ of length $3n$ is put in the common random string and the committer chooses a random $r \leftarrow \{0,1\}^n$ and computes $G(r)$ for $b = 0$ and $G(r) \oplus t$ for $b = 1$, where $G : \{0,1\}^n \to \{0,1\}^{3n}$ is a one-way function based pseudorandom generator. The simulator simply sets $t = G(r_0) \oplus G(r_1)$ and commits by $G(r_0)$ such that it can be opened with any value $b$ by disclosing $r_b$. Instead of using a one-way function based generator as in the original scheme here we use one-way *permutation* based pseudorandom generator, enforcing a unique random string such that a commitment can be opened correctly (as in the quadratic residuosity construction).

**Definition 5 (Equivocal Bit Commitment Scheme).** *A (non-interactive) equivocal bit commitment scheme with unique openings in the common reference string model is a pair $(\mathcal{C}, \mathsf{Com})$ of efficient algorithms such that*

**Unique Opening.** *With overwhelming probability over the choice of $crs \leftarrow \mathcal{C}(1^n)$ there do not exist $(b_0, r_0) \neq (b_1, r_1) \in \{0,1\} \times \{0,1\}^n$ with $\mathsf{Com}(crs, b_0; r_0) = \mathsf{Com}(crs, b_1; r_1)$.*

**Secrecy.** *For any efficient distribution $\mathcal{B}$ on bits the following random variables are computationally indistinguishable:*

- *Let $crs \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$, $b \leftarrow \mathcal{B}(crs_{\mathsf{Com}})$, $r \leftarrow \{0,1\}^n$ and $C \leftarrow \mathsf{Com}(crs, b; r)$. Output $(crs, b, C)$.*
- *Let $crs \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$, $b \leftarrow \mathcal{B}(crs_{\mathsf{Com}})$, $r \leftarrow \{0,1\}^n$ and $C \leftarrow \mathsf{Com}(crs, 1-b; r)$. Output $(crs, b, C)$.*

**Equivocality.** *There exists an efficient algorithm $\mathcal{EQ}$ such that for any efficient distribution $\mathcal{B}$ on bits the following random variables are computationally indistinguishable:*

- *Let $crs \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$, $b \leftarrow \mathcal{B}(crs)$, $r \leftarrow \{0,1\}^n$ and $C \leftarrow \mathsf{Com}(crs, b; r)$. Output $(crs, C, b, r)$.*
- *Let $(crs, C, r_0, r_1) \leftarrow \mathcal{EQ}(1^n)$ and $b \leftarrow \mathcal{B}(crs_{\mathsf{Com}})$. Output $(crs, C, b, r_b)$.*

As sketched before such equivocal commitments can be built from one-way permutations (although each commitment requires a separate part of the common random string):

**Lemma 1 ([DIO98]).** *Non-interactive equivocal bit commitments with unique openings in the common random string model exist if one-way permutations exist.*

### A.3 Construction of Unique Zero-Knowledge System

Our construction follows the one by Damgård [Dam93] closely. Yet some slight changes are necessary, due to the fact that we do not use the homomorphic quadratic residuosity

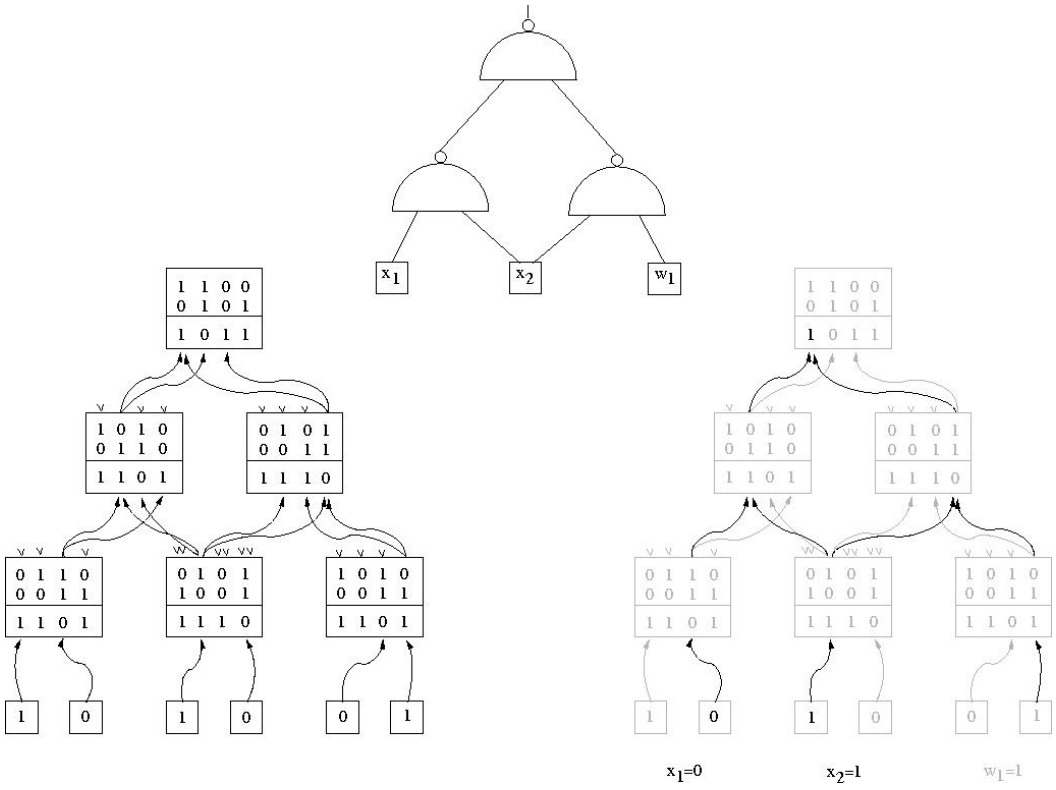


**Figure 4.** Given circuit $\mathtt{C}_n : \{0,1\}^3 \to \{0,1\}$ (top), example construction of "scrambled" circuit with some pointers omitted for sake of readability (lower left corner), and proof through decommitments (dark color) for $x_1 = 0$, $x_2 = 1$ and $w_1 = 1$ (lower right corner).

encryption scheme. As explained above, we only consider the construction of a witness-indistinguishable proof; zero-knowledge follows from the [FLS99] transformation. We write $\text{sUNIZK}_{\mathtt{C}}$ for the following proof system.

We assume wlog. that the circuit $\mathtt{C}_n : \{0,1\}^{\chi(n)} \times \{0,1\}^{\omega(n)} \to \{0,1\}$ consists of binary NAND gates only (any other complete gate works as well, NAND gates are chosen here for sake of concreteness). We also suppose that each of the $G$ gates, among which are $I$ input gates, is enumerated in some fixed order and that no constant gates exists (these can be easily replaced by using appropriate NAND gates for an input bit). We also assume that there is some fixed order in which the circuit is computed, once the input gates are assigned values.

We remark that the construction below is certainly not optimized; this is in order to simplify the description. A toy example for a circuit $\mathtt{C}_n : \{0,1\}^2 \times \{0,1\} \to \{0,1\}$ is given in Figure 4, with two bits $x_1x_2$ for the public input and a single witness bit $w_1$. The reader is invited to consult this example in support of the description below.

*Generating the Common Random String.* The common random string generator $\mathcal{C}(1^n)$ consists of two parts. First it runs the generator $\mathcal{C}_{\mathsf{Com}}(1^n)$ of the equivocal commitment scheme $12G + 8G^2 + 6I$ times to generate strings $crs_{\text{com},i} \leftarrow \mathcal{C}_{\text{com}}(1^n)$. These strings will be divided

by assigning $12 + 8G$ strings to each of the $G$ gates and 6 additional strings to each of the $I$ input gates. The subdivisions for each gate will become clear below. When saying that the prover commits to a bit it is understood that the prover uses the corresponding string in $crs_{\mathrm{com},i}$.

The second part of the common random string consists of a random string $crs_{\mathsf{C}}$ generated for a (possibly ambiguous) non-interactive zero-knowledge proof system $(\mathcal{C}_{\mathsf{C}}, \mathcal{P}_{\mathsf{C}}, \mathcal{V}_{\mathsf{C}})$ to prove correctness of the scrambled circuit (see below). The combined output equals $crs = (crs_{\mathrm{com}}, crs_{\mathsf{C}})$.

*Preparing Keys.* To prepare the public key $PK$ the prover first does the following.

- For each NAND gate $i$ the prover first prepares a $4 \times 3$ truth table $\mathtt{tt}_i$ describing the input/output behavior of a NAND gate, where each input combination 00, 01, 10 and 11 appears in random order. He commits to all bits $\mathtt{tt}_i[a, b]$ for $a \in \{0, 1, 2, 3\}$ and $b \in \{0, 1, 2\}$ individually. We denote by $\mathsf{Com}(\mathtt{tt}_i)$ the committed table. An example might look like this:

| | $\mathtt{tt}_i$ | | | | $\mathsf{Com}(\mathtt{tt}_i)$ | | | |
|---|---|---|---|---|---|---|---|---|
| left input | 0 | 1 | 1 | 0 | $\mathsf{Com}(0)$ | $\mathsf{Com}(1)$ | $\mathsf{Com}(1)$ | $\mathsf{Com}(0)$ |
| right input | 0 | 0 | 1 | 1 | $\mathsf{Com}(0)$ | $\mathsf{Com}(0)$ | $\mathsf{Com}(1)$ | $\mathsf{Com}(1)$ |
| output | 1 | 1 | 0 | 1 | $\mathsf{Com}(1)$ | $\mathsf{Com}(1)$ | $\mathsf{Com}(0)$ | $\mathsf{Com}(1)$ |

- Next, the prover "connects" the two input gates of each gate to the output. That is, if gate $i$ computes the NAND of (left) gate $j$ and (right) gate $k$ then the prover prepares for each output bit $\mathtt{tt}_j[a, 2]$ in the truth table of the left gate $j$ two distinct pointers $\mathtt{pp}_j[i, 0, b]$ for $b \in \{0, 1\}$ (the 0-entry indicates that this is a pointer for a left input).[3] This pointer points to the column numbers in $\mathtt{tt}_i$ matching the input bit of the first row in gate $i$. That is, the output bit $\mathtt{tt}_j[a, 2]$ of gate $j$ equals the input bit $\mathtt{tt}_i[\mathtt{pp}_j[i, 0, 0], 0]$ of gate $i$ and and also $\mathtt{tt}_j[a, 2] = \mathtt{tt}_i[\mathtt{pp}_j[i, 0, 1], 0]$ (see lower left part of Figure 4). The distinct pointers values are arranged in random order and committed to (since the pointer can be described by a number between 0 and 3 one can commit to this pointer bitwise with two bits). The same is also done for the right input gate where we connect according to the second row of gate $i$ and where we use "right" pointers $\mathtt{pp}_k[i, 1, b]$. Since each gate may be connected to other gates as a left or right input (possibly even for the same gate), for each gate $j$ we obtain a $G \times 2 \times 2$ matrix $\mathtt{pp}_j$ of pointers with values from $\{0, 1, 2, 3\}$. We denote by $\mathtt{pp}_j$ this set of pointers of gate $j$ and by $\mathsf{Com}(\mathtt{pp}_j)$ the commitments.
- It remains to specify how to connect the input gates of the circuit. For an input gate $i$ the prover has already prepared a (committed) truth table $\mathsf{Com}(\mathtt{tt}_i)$. Then the prover commits to both input possibilities $\mathtt{vv}_i(b) = b \oplus \$_i$ for a random bit $\$_i$ and associates to each of these two commitments a random pointer $\mathtt{pp}_i(b)$ such that the pointer links the possible input value to the corresponding output value in $\mathtt{tt}_i$.
- Each unspecified matrix entries are set to 0 and committed to.

The public key $PK$ of the prover now consists of all the commitments of the "scrambled" circuit $\mathsf{Com}(\mathsf{C}_n)$, plus a non-interactive zero-knowledge proof by $\mathcal{P}_{\mathsf{C}}$ (with respect to part $crs_{\mathsf{C}}$) that the circuit is formed correctly. Specifically, this means that each truth table describes a NAND gate, that the pointers are connected correctly, and that for each input gate both input possibilities 0 and 1 are connected correctly to the truth table. We denote this proof by $\pi_{\mathsf{C}}$. The prover's secret key $SK$ contains all the decommitments to $\mathsf{Com}(\mathsf{C}_n)$, while $PK = (\mathsf{Com}(\mathsf{C}_n), \pi_{\mathsf{C}})$.

[3] This distinction between left and right pointers is necessary for the case that one gate serves as both inputs to another gate.

*Proving Statements.* To prove a statement $x = x_1 x_2 \dots x_{\chi(n)} \in \{0,1\}^{\chi(n)}$ for witness $w = w_1 w_2 \dots w_{\omega(n)} \in \{0,1\}^{\omega(n)}$ the prover first decommits to the input values in the $x$-part of the input gate $i$ of circuit $\mathtt{C}_n$, i.e., to $\mathtt{vv}_i(b) = x_j$ and to the corresponding pointer $\mathtt{pp}_i(b)$. The prover also reveals the decommitments of the pointers $\mathtt{pp}_i(b)$ of the witness bit $w_j$ (but not the actual values $\mathtt{vv}_i(b) = w_j$). We say that *the corresponding gates have been evaluated to* $\mathtt{tt}_i[\mathtt{pp}_i[b], 2]$.

Then, the prover goes through the computation in the prescribed order such that only evaluated gates serve as inputs to other gates. For each NAND gate $i$ with inputs from left gate $j$ and right gate $k$ (possibly $j = k$) the prover decommits to a pointer pair $\mathtt{pp}_j[i, 0, a]$, $\mathtt{pp}_k[i, 1, a']$ for the input gates $j, k$ such that

- gate $j$ has been evaluated to $\mathtt{tt}_j[a, 2]$ before, i.e., the pointer $\mathtt{pp}_j[i, 0, a]$ is in the same column as the value of the gate $j$, and
- gate $k$ has been evaluated to $\mathtt{tt}_k[a', 2]$ before, i.e., the pointer $\mathtt{pp}_k[i, 1, a']$ is in the same column as the value of the gate $k$, and
- $\mathtt{pp}_j[i, 0, a] = \mathtt{pp}_k[i, 1, a']$, i.e., both pointers point to the same column in the truth table of the output gate $i$.

Finally, the prover opens the commitment of the output gate $i$ for the value $\mathtt{tt}_i[a, 2]$ it has been evaluated to. The proof $\pi$ consists of all decommitments of the pointers, the decommitments of the $x$-input part and of the output gate (see lower right corner in Figure 4).

*Verifying Proofs.* To verify a proof $\pi$ for statement $x$ with respect to $crs = (crs_{\mathrm{com}}, crs_{\mathtt{C}})$ and public key $PK = (\mathsf{Com}(\mathtt{C}_n), \pi_{\mathtt{C}})$, the verifier checks with $\mathcal{V}_{\mathtt{C}}$ that $\pi_{\mathtt{C}}$ is correct with respect to $crs_{\mathtt{C}}$, that the decommitment to the circuit's output bit is a valid decommitment to 1, and that all pointers have been opened correctly, i.e., that each outgoing pointer is in the same column as both incoming pointers. If all tests succeed then $\mathcal{V}$ returns 1.

**Lemma 2.** *Let $(\mathcal{C}_{\mathsf{Com}}, \mathsf{Com})$ be a non-interactive equivocal bit commitment scheme with unique openings in the common random string model, let $(\mathcal{C}_{\mathcal{C}}, \mathcal{P}_{\mathcal{C}}, \mathcal{V}_{\mathcal{C}})$ be a regular NIZK proofs system for $\mathcal{NP}$, and let $\mathcal{C} = (\mathtt{C}_n)_{n\in\mathbb{N}}$ be a sequence of circuits $\mathtt{C}_n : \{0,1\}^{\chi(n)} \times \{0,1\}^{\omega(n)} \to \{0,1\}$. Then the scheme $\mathrm{sUNIZK}_{\mathcal{C}}$ is a single-theorem unique non-interactive witness-indistinguishable proof system for $R^{\mathcal{C}} = \left\{ (x, w) \in \{0,1\}^{\chi(n)+\omega(n)} \;\middle|\; \mathtt{C}_n(x, w) = 1 \right\}$ in the common random string model.*

*Proof.* Completeness is clear. It remains to show soundness, witness indistinguishability and uniqueness.

*Soundness.* For soundness assume that a malicious prover, on input $crs \leftarrow \mathcal{C}(1^n)$, outputs $(PK, x, \pi)$ such that $x \notin L_{R^{\mathrm{c}}}(crs, PK)$ but the verifier accepts. Since the verifier tests the proof $\pi_{\mathtt{C}}$ in the key $PK$ (with respect to $crs_{\mathtt{C}}$) we can assume that the commitments are formed correctly and describe the circuit $\mathtt{C}_n$ (the cheating probability that this is not the case is negligible by the soundness of the regular NIZK). Since the proof $\pi$ must contain the opening to $x$ it follows together with the correctness of the commitments that $\pi$ contains openings to a valid computation of $\mathtt{C}_n(x, w^*)$ for some $w^* \in \{0,1\}^{\omega(n)}$. Because $x \notin L_{R^{\mathrm{c}}}(crs, PK)$, however, any $w^* \in \{0,1\}^{\omega(n)}$ evaluates to 0 and the proof $\pi$ cannot contain a valid 1-decommitment to the circuit's output.

*Witness Indistinguishability.* We show that the distribution of the generated proof does not depend in a significant way on the actual witness. Consider an attacker $\mathcal{D}$ on the witness indistinguishability, getting as input a common random string $crs$ and a key $PK$ generated by the prover. The attacker outputs $(x, w_0, w_1)$ and the prover generates $\pi$ for input $(x, w_b)$ for random bit $b$, and $\mathcal{D}$ tries to predict $b$ through bit $d$. We denote this experiment by $\mathsf{Game}_0(n)$.

In the sequel we condition on the choice of $b = 0$ or $b = 1$. For a fixed bit $b$ consider a slightly modified attack, $\mathsf{Game}_1(n)$, in which instead of generating the circuit's correctness proof $\pi_{\mathsf{C}}$ for random $crs_{\mathsf{C}}$, we now run the zero-knowledge simulator of the regular NIZK to produce the part $crs_{\mathsf{C}}$ of the reference string and a simulated proof $\pi_{\mathsf{C}}$. By the zero-knowledge property of the regular proof system the probability for $d = b$ for fixed $b$ remains negligibly close to the one of the original attack. Hence, for a negligible function $\nu(n)$,

$$\begin{aligned}
\text{Prob}[\, d = b \text{ in } \mathsf{Game}_1(n)] &= \tfrac{1}{2} \cdot \text{Prob}[\, d = b \text{ in } \mathsf{Game}_1(n) \mid b = 0\,] \\
&\quad + \tfrac{1}{2} \cdot \text{Prob}[\, d = b \text{ in } \mathsf{Game}_1(n) \mid b = 1\,] \\
&= \tfrac{1}{2} \cdot (\text{Prob}[\, d = b \text{ in } \mathsf{Game}_0(n) \mid b = 0\,] \pm \nu(n)) \\
&\quad + \tfrac{1}{2} \cdot (\text{Prob}[\, d = b \text{ in } \mathsf{Game}_0(n) \mid b = 1\,] \pm \nu(n)) \\
&= \text{Prob}[\, d = b \text{ in } \mathsf{Game}_0(n)] \pm \nu(n)
\end{aligned}$$

Next, in the slightly changed attack, $\mathsf{Game}_2(n)$, we also modify how the prover works. We assume that, instead of using strings $crs_{\mathsf{Com}} \leftarrow \mathcal{C}_{\mathsf{Com}}(1^n)$, we run algorithm $\mathcal{EQ}$ to generate all strings $crs_{\mathsf{Com}}$ in $crs$ (together with ambiguous openings $r_0, r_1$ for each string). Then we let the prover generate the scrambled circuit as before, but each time the prover is supposed to commit to a bit $a$ we compute this commitment as $\mathsf{Com}(crs_{\mathsf{Com}}, a; r_a)$ for the given values $r_0, r_1$. As in $\mathsf{Game}_1(n)$, the circuit's correctness proof $\pi_{\mathsf{C}}$ is generated by the simulator of the regular NIZK, i.e., without the decommitments. Similar to the previous case it follows from the equivocality that the probability for $d = b$ in this experiment $\mathsf{Game}_2(n)$ is negligibly close to the one in $\mathsf{Game}_1(n)$.

Because of the equivocality of the commitments and the simulation of the circuit's correctness proof the distribution of the prover's data $PK$ and $\pi$ in $\mathsf{Game}_2(n)$ now is independent of the witness. The probability for $d = b$ is thus exactly $1/2$ in this case, and the claim follows for the original attack $\mathsf{Game}_0(n)$.

*Uniqueness.* Finally, we show that the proof above is unique. We claim that the bijection between proofs and witnesses is defined by the commitments and openings of the witness part. That is, with overwhelming probability the revealed pointers of the witness parts (together with the corresponding value) give rise to a mapping $\tau_{crs,PK,x}$ (not necessarily efficiently computable) from proofs to the witnesses.

We call $crs_{\text{com}}$ good if there does not exist a commitment with ambiguous openings. This happens with overwhelming probability and from now on we condition on such $crs_{\text{com}}$. We furthermore presume that the scrambled circuit $\mathsf{Com}(\mathsf{C}_n)$ is correctly formed; this is true for accepted proofs with overwhelming probability (over the choice of $crs_{\mathsf{C}}$) by the validity of the proof $\pi_{\mathsf{C}}$. The latter is also implied with overwhelming probability if the set of accepted proofs $\Pi_{crs,PK}(x)$ is non-empty.

Under the conditions above the openings of the $x$-part and the pointers for the witness part uniquely determine subsequent openings. Hence, any difference in two valid proofs $\pi \neq \pi'$ is due to the opening of the pointers of the witness part. It follows that these proofs describe two distinct witnesses $w \neq w'$ as well. Vice versa, different witnesses entail openings to different pointers and thus to different proofs. □

As explained above, we now apply the technique of [FLS99] to go from a witness-indistinguishable proof to a multiple zero-knowledge proof system. For parameter $n$ assume that the witness length $\omega(n)$ is super-logaritmic (e.g., by padding all witnesses to $n$ bits) and let $\ell(n) = 2 \cdot \omega(n)$. The transformation takes a part $crs_{\mathrm{FLS}} \in \{0,1\}^{\ell(n)}$ of the reference string $crs$ and evaluates the circuit $\mathtt{C}_n^{\mathrm{FLS}} : \{0,1\}^{\chi(n)+\ell(n)} \times \{0,1\}^{\omega(n)} \to \{0,1\}$ defined through

$$\mathtt{C}_n^{\mathrm{FLS}}(x||crs_{\mathrm{FLS}}, w) = \mathtt{C}_n(x, w) \vee [G(w) = crs_{\mathrm{FLS}}]$$

for a pseudorandom generator $G : \{0,1\}^{\omega(n)} \to \{0,1\}^{\ell(n)}$. Note that the input size of the derived circuit slightly enlarges, but only as a function of $n$. Also note that our witness-indistinguishable proof system allows dependencies of theorems $x$ of such bounded parts of $crs$.

For possibly malicious provers the probability that there exist a pre-image of the truly random string $crs_{\mathrm{FLS}}$ under $G$ is at most $2^{-\ell(n)/2}$ and thus negligible, preserving soundness as well as uniqueness. On the other hand, the zero-knowledge simulator can now pick $crs_{\mathrm{FLS}}$ as the pseudorandom value $G(w)$ for random $w$ and then uses this witness $w$ to give proofs. Hence, we obtain:

**Theorem 5.** *Single-theorem unique non-interactive zero-knowledge proof systems for $\mathcal{NP}$ in the common random string model exist if (regular) non-interactive zero-knowledge proof systems for $\mathcal{NP}$ in the common random string model and one-way permutations exist.*

The theorem holds for both bounded and unbounded provers. For efficient provers it follows that sUNIZK proofs exist if trapdoor permutations exist, and for unbounded provers they can be built from any one-way permutation [FLS99].