Round-Optimal GUC-Secure Blind Signatures from Minimal Computational and Setup Assumptions

Michele Ciampi[1], Pierpaolo Della Monica[2], and Ivan Visconti[2]

[1] University of Edinburgh, Edinburgh, UK, michele.ciampi@ed.ac.uk
[2] Sapienza University of Rome, Rome, Italy, {dellamonica,visconti}@diag.uniroma1.it

Abstract. A blind signature scheme is an interactive protocol that enables a user to obtain a signature on a message without revealing any information about the message–signature pair to the signer. Despite more than 40 years of research, all existing constructions suffer from at least two of the following limitations.

1. The protocol is not round-optimal, requiring more than two messages to be exchanged during the signature phase.
2. There is only game-based security and/or lack of composability with global and observable setup (i.e., there is a need for trusted parameters or to program random oracles).
3. Security (unlike regular signatures) is based on demanding hardness assumptions, especially when considering quantum attacks.

In this work, we show how to blindly sign a message, simultaneously overcoming all of the above three limitations. Specifically, we construct a Universally Composable (UC), two-round (optimal) blind signature protocol that relies only on one-way functions (optimal), without trusted parameters. The only deviation[3] from the plain model is the need for a global non-programmable random oracle (NPRO). Nicely, our scheme can be instantiated from a variety of assumptions believed to be post-quantum secure (e.g., AES).

A central technical component of our scheme is the construction of a novel commitment scheme that enjoys a special (mild) form of composability, which may be of independent interest.

We also discuss a concrete instantiation of our scheme that is suitable for practical applications.

1 Introduction

The concept of blindly signing a message was introduced by Chaum [Cha82] as a two-party protocol where a signer, who possesses the secret key, releases a signature to a user who owns the message to be signed and that will remain hidden.

Applications and real-world deployments. Blind signatures are among the most powerful privacy-enhancing technologies. In recent years, they have gained significant attention in industry deployments, including Apple's iCloud Private Relay [App21], Google One's VPN Service [Ara22], Privacy Pass [DGS+18], and various forms of anonymous tokens [HIP+21, Goo20]. Beyond these, blind signatures remain essential for traditional applications such as e-voting systems [FOO93], anonymous credentials [CL01, CG08], and electronic cash [Cha82, OO92]. The Swiss National Bank is currently investigating approaches based on classical work on electronic cash [CGM21] where the central bank takes the role of the signer in the blind signature.

Requirements for real-world usage. Given their centrality in several real-world applications, the security of blind signatures is of great importance. Specifically, in all the aforementioned scenarios, blind signatures represent a crucial building block for larger real-world protocols where multiple signing sessions can occur concurrently and interleave with other operations (e.g., one can spend some coins received from the bank while minting others). Consequently, achieving the security of such primitive when used concurrently with other protocols is essential to guarantee their security in real-world deployments.

Moreover, constructing blind signatures based on well-established generic and minimal assumptions, rather than relying on ad-hoc assumptions (that might be broken and subsequently compromise entire real-world systems), is necessary to ensure confidence, agility and reliability in practical deployments.

Nevertheless, the literature contains several impossibility results for constructing blind signatures [Lin03, FS10, KSY11, Pas11, BL13], implying that blind signatures necessarily require certain types of assumptions (e.g., in the plain model, when considering simulation-based security, concurrent blind signatures are

[3] A deviation is anyway necessary in light of known impossibility results [Lin03].

©IACR 2026. This is the full version of an article that will be published in the proceedings of Crypto 2026.

impossible regardless of the underlying assumptions used [Lin03]). Notice that UC-secure blind signatures are achievable with trusted parameters assuming trapdoor permutations [CLOS02], even round-optimal ones [Fis06a]. Still, it is crucial to avoid constructions built on top of multiple points of failure. This leads us to our first question:

> *Can we construct a* UC-secure *round-optimal blind signature scheme from minimal cryptographic assumptions while relying only on a minimal use of a global setup?*

Furthermore, for long-term deployment, it would be ideal for blind signatures to be post-quantum secure. Simultaneously, we require them to still be efficient enough for practical usage. This motivates our second question:

> *Can such constructions be both based on reliable post-quantum hardness assumptions and sufficiently efficient for real-world deployment?*

We emphasize the focus of conservative and reliable assumptions, especially in light of recent cryptanalytic developments where supposedly post-quantum secure primitives like SIKE [JAC+22] were completely broken [CD23]. Note that, even for widely-studied lattice-based cryptography, which remains a leading candidate for standardization, cryptanalytic attempts continue to be made [Che24, Zha25].

1.1 Our Contribution

We positively answer the above open questions by constructing a 2-round[4] blind signature scheme in Minicrypt. Notably, our construction relies, as its only form of setup, on a global (non-programmable) random oracle (RO) [CJS14, CDG+18]. To the best of our knowledge, ours are the first round-optimal UC secure blind-signatures in Minicrypt. We provide a comprehensive comparison with existing results in Tab. 1, and informally state the main theorem we prove below.

Theorem (informal). *Assuming the existence of OWFs, and assuming as the only form of setup the existence of a non-programmable random oracle (NPRO), there exists a round-optimal Universally Composable blind signature scheme.*

Our construction is not only of theoretical interest. Indeed, its signature size is in the order of magnitude of existing schemes in the post-quantum domain, which are based on stronger complexity assumptions and/or setups[5].

On the minimal use of a global setup. Our objective in this work is to make a minimal setup assumption for proving the security of blind signatures, while still obtaining a construction secure in the UC setting. We first recall that achieving a UC-secure (even only self-concurrent secure) blind signature scheme in the plain model (i.e., without any setup assumption) is impossible [Lin03]. Thus, some form of setup is necessary.

In this work, we prioritize relying on a realistic *global* setup (i.e., a setup in which all protocol instances share the same setup functionality, rather than each instance using its own independent copy, as is usually done in the real world). Concretely, we aim to assume only the classical and widely used existence of a random oracle (RO) that any party can query to obtain an output[6].

In the literature, there are several functionalities modeling a global RO. In particular, Camenisch et al. [CDG+18] provide a detailed treatment, including an *observable* RO (a là [CJS14]), and also a more "powerful" RO that additionally enables *programmability* by the ideal-world adversary[7].

[4] We follow the standard convention in literature and count the number of rounds as the total number of messages exchanged between the parties. That is, in a 2-party protocol, if each party sends exactly one message, we consider it a 2-round protocol.

[5] Note that this comparison holds when our blind signature scheme is instantiated from collision-resistant hash functions (CRHFs) rather than just OWFs.

[6] We aim to avoid assuming a common reference string due to its limited practicality and the fact that it requires a trusted setup procedure in the real world (e.g., assuming that at least one party involved in generating the CRS behaves honestly).

[7] Formally, programmability in the UC setting is more involved: roughly, it provides an interface allowing all parties to set RO answers, and also an interface to detect whether an answer was programmed by another party, with the only exception being when it was programmed by the ideal-world adversary, in which case it looks unnoticeable to other parties. We stress that this idealization is not well aligned with practical heuristic instantiations (e.g., SHA256), where no such programmability is available.

A natural question, then, is whether to allow a programmable RO or to rely only on an observable RO (where the ideal-world adversary can only observe RO queries). We take the latter approach for two reasons. First, programmability of the RO by every party is widely agreed to be misaligned with the real-world heuristic instantiation of an RO (e.g., via SHA256), where no party can program its outputs. Second, and more importantly, Canetti and Chen [CC25] very recently identified weaknesses in existing programmable RO formulations of Camenisch et al. [CDG+18], and proposed an alternative programmable RO functionality. This highlights that programmability of the RO is not yet a well-understood UC setup assumption, and suggests that basing security on such an assumption is not yet reliable or conservative in the UC setting. Accordingly, in this work, we focus on the more challenging task of relying only on an observable RO[8].

Novel commitment scheme. Our blind signatures are based on an extractable commitment scheme that we construct and prove secure under a novel definition that easily allows to maintain hiding and extractability of the commitment in a UC setting where a global random oracle is assumed as a form of setup. In particular, leveraging the use of sessions ids as parts of the inputs to queries to the global random oracle, our scheme is guaranteed to retain the hiding property in a certain session sid even in the presence of an extractor who can extract from adversarially generated commitments in different sessions (identified by session-id different from sid).

Despite our notion not capturing the equivocation property common to almost all full-fledged UC secure schemes, a commitment satisfying our new definition can be naturally used in the design of UC secure protocol where the commitments, once generated, do not need to be opened. Indeed, we observe that in the design of complex UC protocols (including our blind signatures), it can happen that the commitments are generated but never opened. Because we do not need equivocation, we can prove the security of our scheme relying on a global (and non-programmable) random oracle functionality.

An additional feature that we rely on is that our commitment scheme is *provable*, meaning that it is possible to issue proofs related to the content of the commitment. We formalize this by requiring that the opening algorithm of the commitment does not have access to the RO, and this gives us a clean and RO-free way to refer in a relation to the committed message.

Note that non-interactive straight-line extractable schemes that are also *provable* were previously proposed in the literature. Ganesh et al. [GKO+23] propose a construction, but it requires a CRS. In a follow-up work, Badertscher et al. [BCC+24] eliminate the need for a CRS and of programming the RO, but prove their result in a variant of the UC framework (i.e., shielded oracle model).

Theorem (informal). *Assuming the existence of OWFs, and assuming as the only form of setup the existence of an NPRO, there exists a* provable *non-interactive extractable commitment scheme.*

Implications to commit-and-prove protocols. Our notion of non-interactive straight-line extractable and provable commitment schemes has interesting implications also for the literature on commit-and-prove protocols. In such protocols, a prover commits to a value and subsequently proves statements about it without revealing the value. However, existing definitions, such as those by Kiyoshima [Kiy20, Definition 3] and Khurana et al. [KOS18, Definition 4], only enable extraction of the committed value after the proving phase and do not support straight-line extraction from the commitment alone. In contrast, our definition enables efficient, straight-line extraction immediately after the commitment phase, before any proof is provided. This property is essential in settings where extraction from the commitment is required within the security proof, before any statement is proved about the committed value. Consequently, existing commit-and-prove definitions from the literature cannot replace our definition.

2 Technical Overview

We provide a high-level yet detailed overview of our contributions. Although our main goal is to construct a GUC-secure blind signature scheme assuming only OWFs and the existence of a global NPRO, we begin with a core building block required for this construction: a new straight-line extractable commitment scheme (from OWFs) with an additional property, which we call the *provability* property, that is essential for our main result. Although not recommended, readers interested only in the final construction may skip directly to Sec. 2.2 of this overview, which describes the blind signature scheme.

[8] We call this more challenging since, for example, achieving non-interactive zero-knowledge in the observable RO model is clearly impossible, whereas allowing programmability enables many classical constructions.

Table 1: Comparison of recent and/or representative blind signatures across cryptographic assumptions, number of rounds (i.e., messages exchanged between parties), security type, and underlying model. The assumption 2SDH is a strong variant of the SXDH assumption introduced by Okamoto [Oka06]. We also include constructions in the plain model using complexity leveraging techniques. The AGAM represents a variant of AGM for group actions. In the **Security Type** column, "GB" means that the scheme is secure under self-concurrent composition, and considering the Game-Based security definition for blind signature, (†) indicates that the scheme realizes a slightly different UC functionality (i.e., equivocal blind signatures). "UC" means that the scheme is UC-secure, and "GUC" means that the scheme is UC-secure and it relies only on global setup. In the **Underlying Model** column, pro. stands for programmable and non-pro. for non-programmable.

Reference	Cryptographic Assumptions	Protocol Rounds	Security Type	Underlying Model
Tessaro et al. [TZ22]	DL	3	GB	pro. RO+AGM
Crites et al. [CKM+23]	DL	3	GB	pro. RO+AGM
Kastner et al. [KLX22]	DL	3	GB	pro. RO+AGM
Hanzlik et al. [HLM+25]	CSIDH/LESS	3	GB	pro. RO+AGAM
CTZ [CATZ24]	CDH	4	GB	pro. RO
Klooß et al. [KRW24]	DDH	4	GB	pro. RO
Brandt et al. [BHKR25]	DDH	4	GB	non-pro. RO
Klooß et al. [KR25]	DDH	5	GB	pro. RO
Kastner et al. [KNR24]	sRSA+DDH	2	GB	pro. RO
Abe et al. [AJOR18]	SXDH	2	GB	pro. CRS
Garg et al. [GRS+11]	exp-OWF+TDP+DDH	2	GB	–
Garg et al. [GG14]	subexp-DLIN	2	GB	–
Agrawal et al. [AKSY22]	OM-ISIS+NTRU+LWE	2	GB	pro. RO
del Pino et al. [dPK22]	LWE+SIS	2	GB	pro. RO
Beullens et al. [BLNS23]	NTRU+LWE+SIS	2	GB	pro. RO
Jeudy et al. [JS24]	LWE+SIS	3	GB	pro. RO
Bouillaguet et al. [BFM+25]	UOV/Wave	2	GB	pro. RO
Herranz et al. [HL25]	CRHF	2	GB	pro. RO
Kiayias et al. [KZ08]	2SDH	4	UC(†)	pro. CRS
Fischlin [Fis06a]	PKE+ss-NIZK	2	UC	pro. CRS
This work	OWF	2	GUC	non-pro. RO

2.1 Provable, Straight-line Extractable Commitments

Let m be the message that we want to commit to. A common way to construct a straight-line extractable commitment is to commit to m using a standard non-interactive commitment $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$, and then prove via a non-interactive (NI) zero-knowledge (ZK) argument of knowledge (AoK) Π, the knowledge of the committed message. Hiding is guaranteed by the hiding of the commitment and the ZK of Π. Extractability is instead guaranteed via the AoK property of Π. In particular, the final scheme we obtain is straightline extractable (i.e., the extractor does not need to rewind) if the AoK extractor of Π is also straightline extractable. This approach has, unfortunately, a major limitation: NIZK requires a strong form of setup (i.e., at least a common reference string, or a programmable RO).

Our goal is to bypass the above limitation. Our starting point is the *interactive* extractable commitment proposed in [PW09], which we refer to as PW (due to the authors Pas and Wee) and that we now briefly recall. In [PW09], the sender computes two shares s_0, s_1 such that $s_0 \oplus s_1 = \mathsf{m}$, and commits to these shares individually using a non-interactive commitment, thus obtaining τ_0, τ_1 and sends $\mathsf{a}{:=}(\tau_0, \tau_1)$ to the receiver[9]. The receiver picks a random bit b, sends it to the sender, who sends the opening of τ_b, which we denote with z. The extractor for this protocol works as follows. It acts as the receiver using a random bit b, thus obtaining the opening of τ_b (i.e., the extractor obtains the string s_b), then rewinds the sender to obtain the opening of τ_{1-b}, thus obtaining both the strings s_0 and s_1, hence reconstructing the message[10].

Interestingly, the PW protocol has a structure similar to that of a Σ-protocol [Dam10]. In particular, given two accepting transcripts (related to two commitment phases) that share the same first round, but have different second rounds, it is possible to extract the committed message (which can be thought of

[9]For technical reasons this process must be repeated in parallel multiple times but for simplicity we discuss here the main idea only.

[10]Clearly the protocol needs to be executed in parallel $\omega(\log \lambda)$ times to guarantee that the extractor succeeds even against a corrupted sender who may abort.

as a witness in the case of Σ-protocols). This property makes Σ-protocol suitable for the Fiat-Shamir (FS) transform[11]. Our main idea is to apply the FS transform on the PW construction to make the commitment non-interactive. Indeed, the FS transform guarantees that it is possible to extract the message (the witness in the case of Σ-protocols) by allowing the extractor to have full control of the RO (i.e., program the RO) and rewinding the sender (i.e., non-straight-line extraction).

Despite this approach indeed yielding to a non-interactive extractable commitment, our final objective is to have a *straight-line* extractor (that does not program the RO). Hence, we need to follow a slightly different path. In particular, we build upon a variant of the FS transform proposed by Fischlin [Fis05]. At a very high level, our new straight-line extractable commitment scheme works as follows. The sender runs r (where r depends on the security parameter) independent executions of PW, thus obtaining r first rounds denoted with $(\mathsf{a}_1, \ldots, \mathsf{a}_r)$. Then the sender, for every $j \in \{1, \ldots, r\}$ picks a random bit-string c_j, and uses it to complete the j-th PW instance, thus obtaining z_j, such that the tuple $(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)$ is an accepting PW transcript. The sender then checks whether $\mathcal{H}_\ell((\mathsf{a}_1, \ldots, \mathsf{a}_r), j, \mathsf{c}_j, \mathsf{z}_j) = 0$ (where ℓ can be thought as a small parameter, and $\mathcal{H}_\ell$ informally[12] indicates the output of the RO truncated to ℓ bits). If this is the case, then the sender continues; otherwise, it samples a new random c_j, and repeats the process (i.e., the computation of z_j and the RO evaluation for the j-th step). At the end of the r iterations, the sender sends all the tuples $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ to the receiver, and the receiver accepts the commitment if for all $j \in [r]$ the following hold: 1) $(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)$ is an accepting PW transcript and 2) $\mathcal{H}_\ell((\mathsf{a}_1, \ldots, \mathsf{a}_r), j, \mathsf{c}_j, \mathsf{z}_j) = 0$.

Intrinsically, the extractor leverages the fact that the sender, to complete the commitment phase, needs to necessarily query the RO with multiple PW accepting transcripts, all sharing the same first round, but having different challenges. Hence, the extractor, having only the queries made by the sender to the RO, can extract the committed message. This is exactly what makes the extractor straight-line and does not require programming the RO.

After having argued about the extraction property, let us focus on how to prove the hiding property. One could use a similar approach as the one that Fischlin [Fis05] uses to prove that the protocol is ZK when run with a standard Σ-protocol. Following this approach, we would unfortunately need to program the RO. We, however, observe that in our case, we can adopt a simpler strategy. We construct a sequence of hybrid arguments, changing one by one the values committed in each commitment pair (τ_0, τ_1) of the multiple PW executions. We will argue that in the end, we can switch the committed value from a message m_0 to a different message m_1, without the adversary noticing it. We emphasize that adopting this different strategy is crucial: it enables us to achieve our goal of minimal assumptions, whereas naively following Fischlin's approach would not yield the same result. See Sec. 4.2 for the full details.

Instantiation from OWFs. In the description of our straight-line extractable commitment, we leverage a non-interactive commitment. It is well known that non-interactive commitments (in the plain model) can be constructed assuming only 1-to-1 OWFs [MP12] (i.e., standard OWFs are insufficient).

However, this is insufficient for our purposes, as we aim to build a blind signature (leveraging our commitment) based only on OWFs. We address this by observing that the NPRO provides non-interactive commitments from OWFs alone. Specifically, Naor's two-round commitment scheme [Nao91] is based on OWFs and allows the first message (a random string) to be reused. We can thus transform it into a non-interactive commitment by fixing the NPRO output as the first message of Naor's scheme. This enables us to build straight-line extractable commitments assuming only the existence of OWFs. We provide further details in the technical sections.

The main body of the paper focuses on the construction of the straight-line extractable commitment assuming 1-to-1 OWFs. We subsequently discuss how to relax this requirement to standard OWFs using the technique described above.

On the composability of our protocol in the Global RO model. We have sketched how to prove hiding and extractability of our commitment, but we recall that ultimately our goal is to use the commitment to design a UC-secure blind signature scheme in the global (non-programmable) random oracle. Using a commitment scheme that only enjoys hiding and extractability in the UC setting can

[11] In the FS transform, the role of the verifier (receiver) is replaced by the RO, that on input, the first round of the protocol returns a random value that can be used as the second round of the protocol

[12] To give the idea behind our approach and not burden the technical overview for now we use for RO this notation but in the technical part of the paper we will use a more explicit notation related to UC setting of the NPRO [CJS14, CDG+18].

lead to problems. This is because there are multiple sessions and the adversary can take the role of both sender and receiver (in different sessions). Formally, this means that in the security proof we may need to rely, at the same time, on the hiding property (against a corrupted receiver) and on the extractability feature (against a corrupted sender) of our commitment scheme. We prove that our scheme is secure in such a setting. In particular, we prove that the hiding of the commitment issued in a given session is maintained, even if the extractor is executed concurrently in different sessions.

We formalize this feature in a novel yet natural way. We slightly modify the standard definitions of hiding and extractability we sketched above as follows. The hiding security game is now parameterized by a session identifier sid, which is used as an additional parameter of the commitment scheme. Moreover, we explicitly require the parties to have access to a global non-programmable random oracle a là [CJS14, CDG+18]. In this, the adversary can request to see the queries made to the RO of the form $(\mathsf{sid}', \mathsf{sid}||x)$ such that $\mathsf{sid}' \neq \mathsf{sid}$. Following [CDG+18], we call these queries *illegitimate*.

In summary, we require hiding to hold even against such a stronger adversary, who can see the queries made in other sessions.

The definition of extractability follows accordingly. The security experiment is again parameterized by a session identifier sid, and the extractor can get access to all the queries issued by the adversarial sender of the form $(\mathsf{sid}, \mathsf{sid}||x)$, and all the illegitimate queries. In summary, the extractor can access all the queries of the form $(\mathsf{sid}', \mathsf{sid}||x)$ for any sid' (i.e., the only queries not seen by the extractor are those of the form $(\mathsf{sid}, \mathsf{sid}||x)$ made by honest senders).

The above syntactical change in the definitions of hiding and extractability makes the commitment quite easy to use in a composable setting. In particular, consider the case where we have a corrupted sender in a session $\mathsf{sid}_{\mathsf{EXT}}$ and a corrupted receiver in session $\mathsf{sid}_{\mathsf{HDN}}$. Our definition allows us to rely on the hiding of the commitment in the session $\mathsf{sid}_{\mathsf{HDN}}$, while we run the extractor on the session $\mathsf{sid}_{\mathsf{EXT}}$. Indeed, the extractor only needs to know the queries of the form $(\mathsf{sid}', \mathsf{sid}_{\mathsf{EXT}}||x)$ for any sid'. The crucial observation is that, as long as $\mathsf{sid}_{\mathsf{HDN}} \neq \mathsf{sid}_{\mathsf{EXT}}$, then hiding of the scheme is defined to hold even against an adversary that knows these queries. Hence, we can rely on the hiding of the scheme in session $\mathsf{sid}_{\mathsf{HDN}}$ while running the extractor in session $\mathsf{sid}_{\mathsf{EXT}}$.

To summarize, we have now constructed a non-interactive commitment scheme straightline extractable using a NPRO.

What about simpler schemes? A (very simple) way to obtain the same result consists of using the folklore approach of committing to a message m, by sampling a random value r and computing com as output of $\mathcal{H}(m\|r)$. This commitment is straight-line extractable, as the extractor just needs to observe what queries the sender makes to the RO to extract the message.

However, this scheme has a well-known limitation: it is not possible to make a proof about the content of the commitment, as this would require knowing the circuit of the RO. Our scheme instead has this property, which we call *provability*. Indeed, to prove something about the message committed in our protocol, one simply needs to issue a proof of correct opening, which does not involve querying the RO (i.e., the opening is only related to the content of the non-interactive commitment generated as part of the first rounds of the PW construction, namely the commitments contained in the tuple $(\mathsf{a}_1, \ldots, \mathsf{a}_r)$).

In our application, we leverage exactly this property by using a non-interactive witness-indistinguishable (WI) AoK, which can be instantiated without assuming the *programmability* of the RO. This is what will ultimately make our signature scheme provably secure without relying on the programmability of the RO. We refer to the technical part in App. A.1 for more details.

Now that we have our main tool, we are ready to describe how our round-optimal blind signature scheme works.

2.2 Blind Signatures in the NPRO from OWFs

Our main tools are the extractable commitment we have just designed, which we denote with ExtCom, a standard signature scheme Sig, and a non-interactive witness indistinguishable AoK, which we denote as Niwi. Our construction at a high-level follows Fischlin's approach [Fis06a] for round-optimal blind signatures.

During the key generation phase, we first obtain a (secret-public) key pair $(\mathsf{sk}', \mathsf{pk}')$ for the signature scheme Sig, and then commit via ExtCom to sk', thus obtaining $\tau_{\mathcal{S}}$. The secret key of our blind signature scheme simply corresponds to sk', while our public key is the pair $(\mathsf{pk}', \tau_{\mathcal{S}})$.

First round. In the first round, the user, on input a message m, commits to m via ExtCom, thus obtaining $\tau_{\mathcal{U}}$, and sends it to the signer.

Second round. The signer, upon receiving $\tau_{\mathcal{U}}$, signs it using the standard signature scheme Sig, thus obtaining σ', and issues a proof leveraging Niwi proving that either 1) he knows how to open $\tau_{\mathcal{U}}$ *OR* 2) he knows the secret key corresponding to pk', and that such a key is committed in $\tau_{\mathcal{S}}$ (that we recall is part of the public key). The honest signer will use the second branch to issue the proof.

Signature computation. When the user wants to issue the signature of the message m, he clearly cannot send σ' in the clear, otherwise this could violate the blindness property. Instead, the user issues a proof leveraging Niwi proving that either 1) there exists a signature σ' valid for the verification key pk' of a commitment $\tau_{\mathcal{U}}$ that opens to the message m *OR* 2) that he knows the secret key sk' for Sig. The honest user will use the first branch, computing a proof π that represents our final signature σ.

To verify if the signature σ (i.e., the proof π) is accepting, one simply needs to verify if the proof π is accepting with respect to the statement $(\mathsf{pk}, \mathsf{m})$.

At a very high level, unforgeability is guaranteed from the unforgeability of Sig, while blindness is guaranteed by the witness indistinguishability of the NIWI proof. To see why our protocol can be proven secure in the NPRO, we note that our simulator needs to perform two main tasks: 1) If the signer is corrupted, then the simulator needs to extract the secret key of Sig that will be sent to the ideal functionality to issue the ideal-world signatures[13]; 2) When the user is corrupted instead, the simulator needs to extract the message m, and then query the ideal functionality with such a message.

In our protocol, both sk and m are committed via our extractable commitment, hence the simulators will be able to extract such information relying on the extractor of ExtCom, which by definition is straight-line and does not program the RO. Some care is needed to prove that the simulators are successful with high enough probability in these tasks, and that the protocol is UC secure in the global RO setting.

We have glossed over the fact that we need a stronger notion of NIWI, which prevents non-malleability attacks. We will formally define the security required by our NIWI in a similar fashion as for our extractable commitment, and then show a simple instantiation based on OWFs in the NPRO. For more details, we refer to the technical part in Sec. 3.1.

Commitments with extractable proofs are not enough. In the technical overview, we presented our commitment scheme and showed how it enables UC-secure blind signatures from OWFs in the NPRO model. We emphasize that our specific construction is essential and cannot be replaced by standard alternatives. In particular, combining a plain-model commitment with a non-interactive straight-line extractable proof of knowledge does not suffice to achieve our lower bounds; we explain why in App. L.

Practical instantiation from CRHFs. We provide an asymptotic analysis of the efficiency of our construction when instantiated via CRHFs. This matches the minimal assumption required in our setting, since the RO must be instantiated in any case. Details on instantiation of the involved cryptographic components, as well as a description of our analysis, are given in App. K. Here we show a schematic comparison in Tab. 2. Notably, our scheme is the only post-quantum secure scheme that is not based on lattice-based or non-standard assumptions.

2.3 Related Works

Foundations of blind signatures. Blind signatures were introduced by Chaum in 1982 [Cha82] in the context of electronic cash. Subsequent early works include those by Juels, Luby and Ostrovksy [JLO97], Pointcheval and Stern [PS00], and Fischlin [Fis06a]. Juels et al. treated blind signatures as a special case of two-party computation and provided a theoretical generic (though inefficient) construction in the plain model based on trapdoor permutations. Pointcheval and Stern demonstrated the security of efficient three-move blind signatures in the RO model, provided that only polylogarithmically many signatures are issued concurrently. In 2003, Lindell [Lin03] established a fundamental limitation by proving the impossibility of achieving concurrently secure blind signatures in the plain model. Lindell's impossibility result motivated Fischlin's seminal work [Fis06a], which introduced a generic construction for round-optimal (two-move) blind signatures in the CRS model to circumvent Lindell's limitation. Fischlin's construction relies on non-interactive zero-knowledge (NIZK) proofs and public-key encryption, and notably achieves security in the Universal Composability (UC) framework [Can20]. Since then, several works [FS10, KSY11, Pas11, BL13] have further constrained the design space for blind signatures under minimal assumptions. Particularly relevant is the result of Katz et al. [KSY11], which ruled

[13]More formally, the simulator uses the secret key to create the algorithm BlSig which, as discussed at the beginning of this section, the functionality requires during the key generation phase.

Table 2: Comparison of potentially practical PQ round-optimal blind signatures. The assumptions related to lattice problems are over polynomial rings. The (†) indicates reliance on the OM-ISIS assumption that stands for the One-More-ISIS assumption. Such an assumption has been introduced by Agrawal et al. in the same paper [AKSY22] to prove the security of their scheme. UOV and Wave stand for the choice of directly assuming the security of UOV/Wave [KPG99, BCC+22] signatures directly, which is an interactive assumption. The (‡) indicates reliance on the paradox heuristic assumption that a hash function can simultaneously be treated as a RO and as an explicit circuit. In the **Model Assumption** column, "GB" means that the scheme is secure under self-concurrent composition, and considering the Game-Based security definition for blind signature, moreover pro. stands for programmable and non-pro. for non-programmable.

Reference	**Hardness Assumptions**	**Signature Size**	**Standard Assumption**	**Model Assumption**
[AKSY22]	OM-ISIS+NTRU+LWE	45.19 KB	✗(†)	GB + pro. RO
[dPK22]	LWE+SIS	102.6 KB	✓	GB + pro. RO
[HL25]	CRHF	$\sim$ 250 KB	✓	GB + pro. RO
[BLNS23]	NTRU+LWE+SIS	22 KB	✗(‡)	GB + pro. RO
[BFM+25]	UOV/Wave	8/90 KB	✗(‡)	GB + pro. RO
This work	CRHF	$\sim$ 250 KB	✓	UC + non-pro. RO

out black-box constructions of blind signatures from one-way permutations, regardless of concurrency considerations.

Efficient blind signatures from strong assumptions. There exists several blind signature schemes relying on complexity leveraging [GRS+11, GG14] or non-standard q-type or interactive assumptions [Oka06, FHKS16, Gha17]. There are also blind variants of BLS signatures and RSA-full-domain hash signatures [Bol03, BNPS03], which are very efficient and round-optimal but rely on interactive one-more variants of the underlying assumptions (e.g., one-more CDH). Very efficient blind signatures in pairing-free groups have been proposed [FPS20, KLR21, TZ22, CKM+23] but rely on the algebraic group model (AGM) [FKL18] and the RO model.

Fischlin and its (PQ) descendants. In parallel with the above approaches, a second line of work has focused on practical instantiations of Fischlin's framework for achieving round-optimal blind signatures while avoiding the strong AGM. This framework has been employed in [KRS23, KNR24] to achieve simple and efficient blind signature schemes based on pairings and the strong RSA assumption respectively, though these are not as efficient as AGM-based approaches in terms of signature size and communication complexity.

Recent works by Agrawal et al. [AKSY22], del Pino and Katsumata [dPK22], and Beullens et al. [BLNS23] have produced the first practical PQ blind signatures by leveraging lattice-based assumptions. Specifically, Agrawal et al. [AKSY22] introduce a new one-more assumption for SIS combined with NTRU and LWE, while del Pino and Katsumata [dPK22] rely on a combination of LWE and SIS assumptions. Similarly, Beullens et al. [BLNS23] based their construction on NTRU together with LWE and SIS. These works share three common characteristics: they all (1) follow Fischlin's framework, (2) are in the programmable RO model, and (3) rely on lattice-based assumptions. Moreover, both [AKSY22] and [BLNS23] make the heuristic assumption that the RO can be instantiated, which is theoretically questionable. In the realm of lattice assumptions, we also highlight that recently Jeudy and Sanders [JS24], relying only on LWE and SIS assumptions, achieve very good performance in terms of signature size at the cost of not having a round-optimal construction (i.e., their construction is 3-round). Beyond lattice-based constructions, the landscape of PQ blind signatures has seen limited success. An early attempt by Petzoldt et al. [PSM17] proposed a multivariate blind signature scheme that was later shown to be insecure by Beullens [Beu25]. Similarly, the isogeny-based construction of [KLLQ24] failed to provide concurrent security. Blazy et al. [BGSS17] introduced a code-based blind signature following Fischlin's approach using NIZK for concatenated matrices, though with signature sizes in the megabyte range. We also note that Tanuki [HLM+25] recently achieved efficient blind signatures based on isogeny-based (CSIDH) and code-based (LESS) assumptions, although this work leverages the AGAM (algebraic group action model), a variant of the AGM, does not follow Fischlin's framework, and does not yield round-optimal signatures.

Finally, a very recent concurrent work by Herranz et al. [HL25] shows that, using classical tools, it is possible to realize Fischlin's construction based on CRHFs.

However, in their original version, their analysis was only sketchy, relied on a programmable RO, and based security on heuristic assumptions for the commitment scheme. After our work appeared online, they updated their version, improving/clarifying their analysis. Still, their updated analysis does not address general composition (i.e., UC security), relies on a programmable RO, and pursues a different objective, focusing primarily on implementation aspects. We report the comparison with their updated result in Tab. 1.

Efficient blind signatures from conservative assumptions. Very recently [CATZ24] aims to achieve efficient blind signatures while relying on well-established conservative assumptions, that is, avoiding pairings, one-more variants, and the AGM. They achieved a blind signature scheme based only on the CDH assumption in the RO model, although it provides a weaker variant of unforgeability and produces signatures of large size, in contrast to the short-size signature of AGM-based blind signatures. This approach was subsequently improved by [KRW24, BHKR25], which rely on DDH and reduces the signature size, ultimately culminating in the recent work of Kloß and Reichle [KR25], which, assuming only DDH, achieves a signature size in the order of a few hundred bytes with a 5-round protocol in the RO model. It is worth noting that, unlike Fischlin's original construction, none of the aforementioned constructions achieve UC security. Furthermore, while significant effort has been made toward minimizing assumptions in the (*programmable*) RO model for blind signatures from non-PQ assumption, at the price of increasing the round-complexity, comparable progress in the post-quantum setting remains limited. Additionally, the assumptions used in recent constructions are still *algebraically structured* (number-theoretic, lattice-based, or code-based) rather than generic, in contrast to the initial work of Juels, Luby, and Ostrovsky.

3 Preliminaries

Standard definitions and notations are provided in App. A. Our work is in the Universal Composability (UC) framework [Can20] and we provide the preliminaries in App. B. We give here the formal definition of the NPRO functionality in Fig. 1, following the functionality of Canetti et al. in [CJS14].

Global observable RO model. Canetti, Jain, and Scafuro in [CJS14] apply the GUC framework to model globally accessible non-programmable random oracles. To do so they define a global RO functionality $\mathcal{G}_{\mathsf{oRO}}$ that allows the simulator to observe certain queries made outside the current session. That is, some of these queries can be marked as "illegitimate" and potentially leaked to the simulator. We recall the formal definition of the $\mathcal{G}_{\mathsf{oRO}}$ functionality in Fig. 1.

Functionality $\mathcal{G}_{\mathsf{oRO}}$

Parameters: Output size μ (it depends on the security parameter λ).
Variables: Initially empty list List.

1. `Query`. On input $(\mathsf{HashQuery}, \mathsf{m})$ from a party $(\mathcal{P}, \mathsf{sid})$ or from the adversary:
   - Find h such that $(\mathsf{m}, h) \in \mathsf{List}$.
     If no such h exists: sample $h \leftarrow\!\!\$\ \{0,1\}^\mu$ and set $\mathsf{List} = \mathsf{List} \cup \{(\mathsf{m}, h)\}$.
   - Parse m as (s, m').
   - If this query is made by the adversary , or if $s \neq \mathsf{sid}$, then add (s, m', h) to the (initially empty) list of illegitimate queries $\mathcal{Q}_s$.
   - Output $(\mathsf{HashConfirm}, h)$ to the functionality caller.
2. `Observe`. On input $(\mathsf{Observe}, \mathsf{sid})$ from the adversary:
   - If $\mathcal{Q}|_{\mathsf{sid}}$ does not exist yet, then set $\mathcal{Q}_{\mathsf{sid}} = \varnothing$.
   - Output $(\mathsf{ListObserve}, \mathcal{Q}_{\mathsf{sid}})$ to the adversary.

Fig. 1: The functionality $\mathcal{G}_{\mathsf{oRO}}$ of the global observable RO.

3.1 Extractable Commitment Scheme

We defer to App. A.1 standard definition of a (3-round) commitment scheme, and we directly focus here on extractable commitments. Informally, a commitment is extractable if there exists an efficient extractor

$\mathsf{Com.Ext}$ that, given black-box access to any efficient malicious sender $\mathcal{A}$ which successfully performs the commitment phase, can output the unique committed message that can be successfully decommitted.

Definition 1 ([PW09, GLOV12] Extractable Commitment Scheme). *A commitment scheme* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ *for a message space* $\mathcal{M}$ *(where* $\perp \notin \mathcal{M}$*) is extractable if there exists an (expected)* PPT *extractor* $\mathsf{Com.Ext}$ *that, given oracle access to any malicious sender* $\mathcal{A}$*, outputs a pair* (τ, m^*) *such that the following properties hold:*

- (**Simulatability**) *The simulated view* τ *is identically distributed to the view of* $\mathcal{A}$ *(when interacting with an honest* $\mathsf{Com.Rec}$*) in the commitment phase.*
- (**Extractability**) *The probability that* τ *is accepting and* $\mathsf{m}^* = \perp$ *is negligible and it is computationally infeasible to open* τ *to any value other than* m^**.*

There also exist extractable commitment schemes that are statistically binding. In this case, the definition is as above, except that if $\mathsf{m}^* \neq \perp$, we require it to be statistically impossible to open τ to any value other than m^*.

Extractable commitment from any commitment scheme. We recall that there exists a protocol that, starting from any commitment scheme $\mathsf{Com}'' = (\mathsf{Com.Sen}'', \mathsf{Com.Rec}'')$, constructs an extractable commitment scheme $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$. This protocol has been widely used in the literature [Kil88, DDN00, PRS02, Ros04, PW09, GLOV12, CCLY22]. We describe the construction in Fig. 2.

Following Def. 1, the proof of hiding and binding can be found in [PW09, Section 4]. The extractor can simply run as a receiver, and if any of the θ commitments is not accepting, it outputs $\mathsf{m}^* = \perp$ (i.e., the commitment transcript τ is not well-formed). Otherwise, it rewinds (Step 2) and changes the challenge until another θ well-formed (partially) decommitments are obtained. Then it verifies that for each decommitment, the XOR of all pairs corresponds to the same string. Then the extractor can extract a value from the responses of these two distinct challenges. The extractor, by playing random challenges in each execution of Step 2, it perfectly simulates the behavior of the receiver, and the analysis in [PW09] shows that its running time is (expected) polynomial.

Let $\mathsf{Com}'' = (\mathsf{Com.Sen}'', \mathsf{Com.Rec}'')$ be a commitment scheme with the message space $\{0,1\}^m$. We describe an extractable commitment scheme $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$, with message space $\{0,1\}^m$.

Common input: Security parameter λ, number of pairs θ to commit to and message length m.

Commitment phase.

1. $\mathsf{Com.Sen}'$: on input $\mathsf{m} \in \{0,1\}^m$, the message to commit to, generates θ random strings $\{s_i^0\}_{i\in[\theta]}$ of the same length as m, and computes $\{s_i^1 = \mathsf{m} \oplus s_i^0\}_{i\in[\theta]}$, therefore $\{\mathsf{m} = s_i^0 \oplus s_i^1\}_{i\in[\theta]}$. Then leveraging Com'' the θ pairs $\{s_i^0, s_i^1\}_{i\in[\theta]}$ are committed. That is, $\mathsf{Com.Sen}'$ and $\mathsf{Com.Rec}'$ produce $\{\tau_i^0 = \langle \mathsf{Com.Sen}''(s_i^0), \mathsf{Com.Rec}'' \rangle, \tau_i^1 = \langle \mathsf{Com.Sen}''(s_i^1), \mathsf{Com.Rec}'' \rangle\}_{i\in[\theta]}$.
2. $\mathsf{Com.Rec}'$: outputs a random θ-bit challenge string $(b_1 \| \ldots \| b_\theta)$.
3. $\mathsf{Com.Sen}'$: decommits the $\{\tau_i^{b_i}\}_{i\in[\theta]}$ commitments (i.e., non-interactively opens θ of previous commitments, one per pair)[a].

Decommitment phase.

1. $\mathsf{Com.Sen}'$: sends m and non-interactively decommits the other θ commitments $\{\tau_i^{\bar{b}_i}\}_{i\in[\theta]}$, where $\bar{b}_i = (1 - b_i)$.
2. $\mathsf{Com.Rec}'$: checks that all openings are valid and that all θ pairs of random strings $\{s_i^0, s_i^1\}_{i\in[\theta]}$ satisfy $\{\mathsf{m} = s_i^0 \oplus s_i^1\}_{i\in[\theta]}$. If so, it considers the value committed to be m, otherwise $\perp$.

[a] The commitment is well-formed (i.e., accepted by the receiver) only if all the openings are valid.

Fig. 2: Construction of an extractable commitment from any commitment scheme. Extraction succeeds with all but negligible probability as long as $\theta = \omega(\log \lambda)$, see [CCLY22].

Our definition of non-interactive extractable commitment scheme in the NPRO. The definition of non-interactive extractable commitment schemes naturally follows from Def. 1. Specifically, in this work, we focus on the notion of non-interactive *straight-line* extractable commitment schemes in the NPRO model, following the one introduced by Pass in [Pas03, Section 4.1].

We provide here the definition of straight-line extractable commitments in the NPRO setting. Specifically, we present a definition for such commitments that accounts for their usage when both the sender and receiver interact with the global functionality $\mathcal{G}_{\mathsf{oRO}}$ from [CJS14], which we recall in Fig. 1. In our definition, we explicitly include a (session) identifier id as input to both algorithms.

Note that our definition is deliberately more detailed to explicitly address the application of these commitments in composable settings. The motivation for this approach is to formalize the property that a simulator in the composability framework can leverage such commitments, achieving hiding in some sessions and straight-line extractability in others. Note that this definition can also be used for non-composable use cases, as the identifier id can simply be ignored (i.e., set to a fixed value). Formally, a non-interactive straight-line extractable commitment scheme is defined by two non-interactive algorithms $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$, where both algorithms have access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality, work as a non-interactive commitment scheme, and satisfy the correctness, hiding and straight-line extractability properties as defined below.

Definition 2 (Correctness). *A non-interactive straight-line extractable commitment scheme* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$*, where both algorithms have black-box access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality, is* perfectly correct *if, for every* $\lambda \in \mathbb{N}$*,* $\mathsf{m} \in \mathcal{M}$*, and* $\mathsf{id}, \mathsf{id}' \in \{0,1\}^*$*, the following holds:*

$$\Pr\big[\mathsf{Com.Rec}(\mathsf{id}', \mathsf{d}, \tau, \mathsf{m}) = 1 \wedge \mathsf{id} = \mathsf{id}' \;\big|\; (\tau, \mathsf{d}) \leftarrow \mathsf{Com.Sen}(\mathsf{id}, \mathsf{m})\big] = 1$$

Definition 3 (Hiding). *Consider the* HDN *game in Fig. 3, parameterized by the security parameter* λ *and a string* $\mathsf{id} \in \{0,1\}^*$*. A non-interactive straight-line extractable commitment scheme* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ *is* computationally hiding *if, for every* PPT *adversary* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}}(\mathcal{A}, \lambda)$ *is negligible in* λ*, where:* $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}}(\mathcal{A}, \lambda) := \big|\frac{1}{2} - \Pr\big[\mathrm{HDN}^{\mathcal{A}}_{\mathsf{Com}}(\lambda, \mathsf{id}) = 1\big]\big|$*, and the algorithms and the adversary* $\mathcal{A}$ *have access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

Game $\mathrm{HDN}^{\mathcal{A}}_{\mathsf{Com}}(\lambda, \mathsf{id})$

1. On input λ and id, the adversary $\mathcal{A}$, outputs $(\mathsf{m}_0, \mathsf{m}_1) \in \mathcal{M}$ of equal length.
2. A uniformly random bit b is sampled, i.e., $b \leftarrow\!\!\$\ \{0,1\}$.
3. The algorithm $\mathsf{Com.Sen}$, on input id and m_b, outputs (τ, d), i.e., $(\tau, \mathsf{d}) \leftarrow \mathsf{Com.Sen}(\mathsf{id}, \mathsf{m}_b)$.
4. $\mathcal{A}$, on input the commitment τ and id, outputs a bit b'.
5. The game outputs 1 if $b = b'$, and 0 otherwise.

Fig. 3: The HDN game. Note that $\mathcal{A}$ having access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality can interact with the `Observe` interface.

Definition 4 (Straight-Line Extractability). *Consider the* SLE *game in Fig. 4 parameterized by the security parameter* λ *and a string* $\mathsf{id} \in \{0,1\}^*$*. A non-interactive straight-line extractable commitment scheme* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ *satisfies the straight-line extractability property if, a) there exists a* PPT *extractor* $\mathsf{Com.Ext}$ *and b) for every* PPT *adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathrm{sle}}_{\mathsf{Com}}(\mathcal{A}, \lambda)$ *is negligible in* λ *where* $\mathsf{Adv}^{\mathrm{sle}}_{\mathsf{Com}}(\mathcal{A}, \lambda) := \Pr\big[\mathrm{SLE}^{\mathcal{A}}_{\mathsf{Com}}(\lambda, \mathsf{id}) = 1\big]$*, and the algorithms and the adversary* $\mathcal{A}$ *have access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

Game $\mathrm{SLE}^{\mathcal{A}}_{\mathsf{Com}}(\lambda, \mathsf{id})$

1. On input 1^λ and id, the adversary $\mathcal{A}$ outputs the tuple $(\mathsf{d}, \tau, \mathsf{m})$.
2. On input the commitment τ, the query-answer list $\mathcal{Q}$ outputted by a call to the `Observe` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality, the extractor $\mathsf{Com.Ext}$ outputs a message m'.
3. The game outputs 1 if: **(a)** the message m is different from m', and **(b)** the algorithm $\mathsf{Com.Rec}$ on input id, the decommitment information d, the commitment τ and the message m outputs 1. Otherwise, the game outputs 0.

Fig. 4: The SLE game. Note that $\mathcal{A}$ having access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality can interact with the `Observe` interface.

Malleability attacks. We highlight that we have carefully developed this definition to explicitly manage queries to the $\mathcal{G}_{\mathsf{oRO}}$ global functionality and introduced id because we anticipate that by properly leveraging these identifiers in accordance with queries to the $\mathcal{G}_{\mathsf{oRO}}$ global functionality, it is possible to construct schemes that safely enable the hiding property in certain sessions (i.e., against malicious receivers) while correctly extracting in other sessions (i.e., against malicious senders). This mitigates potential attacks where a malicious receiver might exploit the extractability of a commitment in one session to break the hiding of a commitment in another, and vice versa.

In App. A.2, we discuss why the commitment scheme of Pass [Pas04], when carefully instantiated according to our definitions and interacting with the global $\mathcal{G}_{\mathsf{oRO}}$ functionality, achieves such an interesting property for composability.

Non-Interactive Witness Indistinguishability systems. In App. A.5, we formally define a non-interactive witness indistinguishability (NIWI) argument of knowledge (AoK) system in the NPRO model in a similar fashion as for our definition of non-interactive straight-line extractable commitment (i.e., we explicitly consider ids and the interaction with the $\mathcal{G}_{\mathsf{oRO}}$ functionality), and then show a simple instantiation based only on the existence of a NPRO.

4 Non-Interactive Straight-Line Extractable and "Provable" Commitments

In [GKO+23], Ganesh et al. informally introduced the concept of "provable" commitment schemes. They define a non-interactive commitment as provable (and straight-line extractable) if it admits a meaningful circuit representation, making it feasible to prove that a value has been committed correctly. This allows to leverage the circuit representation of either the commitment or decommitment algorithms (e.g., as a statement of a NIZK) to prove properties of the committed message while supporting straight-line extraction of this message via the NPRO. Note that if there is no need for straight-line extraction in the NPRO, then standard commitments (e.g., Naor's one) already satisfy this notion. While this kind of commitment might seem straightforward to achieve, we observe that, following [GKO+23], the simple straight-line extractable commitment of Pass [Pas04] described in App. A.1 (i.e., where a single query to the $\mathcal{G}_{\mathsf{oRO}}$ functionality `Query` interface is done both to commit and decommit) has limitations. In this scheme, m is the message to commit and ρ represents the random coins. However, as both the commitment and decommitment algorithms crucially rely on the RO, there is no opportunity to prove statements about the committed message m.

This issue could theoretically be addressed by assuming that the $\mathcal{G}_{\mathsf{oRO}}$ functionality can instantiated with a concrete hash function having a circuit representation (e.g., this is a common assumption in the literature on recursive SNARKs [BCMS20, COS20]). However, following [GKO+23], we avoid such heuristics. The classical result of Canetti, Goldreich, and Halevi [CGH98] proves that one can build (contrived) cryptographic schemes that are provably secure in the RO model but completely broken when the oracle is "instantiated" with any hash function. More importantly, recent work [KRS25] strengthens this result (shifting from theory to practice) by showing that, for every choice of hash function instantiating the RO, there exist attacks against the soundness of a standard and widely used succinct argument system based on the GKR protocol [GKR15].

In [GKO+23], the authors informally define the non-interactive straight-line extractable and provable commitment schemes, mainly focusing on their specific construction of a commitment scheme that allows proving statements about the committed value while maintaining binding and hiding properties, as well as straight-line extractability. We advance this direction by formalizing such a property for commitment schemes and providing a definition building on Def. 4.

Definition 5 (Non-Interactive Straight-Line Extractable and Provable Commitment Scheme). *Assuming the $\mathcal{G}_{\mathsf{oRO}}$ functionality and that the algorithms can interact through its interfaces. A non-interactive straight-line extractable commitment scheme, defined by a pair of* PPT *algorithms* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ *where hiding as in Def. 3 and straight-line extractability Def. 4 hold, is also* provable *if the decommitment algorithm* $\mathsf{Com.Rec}$ *has no access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality.*

4.1 Construction

We propose a black-box construction that, starting from any non-interactive commitment scheme[14], yields a straight-line extractable and provable commitment scheme according to Def. 5. The resulting commit-

[14]The only commitment schemes that are excluded are those that rely on the RO, such as the one of Pass described in App. A.1.

ment is non-interactive, and its security is established in the NPRO model. We present our construction in Fig. 5. Our starting point for building such commitments is the standard extractable commitment scheme of [PW09] described in Fig. 2. The basic idea is to leverage the commitment scheme in Fig. 2 and transform it, using the NPRO, into a non-interactive and straight-line extractable scheme that is also provable, according to Def. 5. Specifically, our transformation is inspired by Fischlin's work [Fis05], where a similar transformation is applied to convert a Σ-protocol (i.e., a 3-round protocol where the prover speaks first, followed by the verifier, and finally the prover again, generating a transcript $(\mathsf{a}, \mathsf{c}, \mathsf{z})$) into a straight-line extractable NIZK in the (programmable) RO[15].

Note that, starting from the extractable commitment scheme described in Fig. 2, our goal is to achieve a non-interactive commitment. Step 1 in Fig. 2 of the commitment phase makes black-box use of any commitment scheme, thus we must restrict ourselves to non-interactive commitment schemes at this point. With this restriction, the resulting extractable commitment in Fig. 2 scheme becomes a 3-round protocol for the commitment phase, where we apply the same "proof of work" idea of Fischlin to make it non-interactive, leveraging the RO.

Rationale behind the construction. That is, given $\ell, r, \theta \in \mathbb{N}^+$ such that $r\ell = \lambda$ and $r > 1$, and a string $\mathsf{id} \in \{0,1\}^*$, we build a commitment scheme that informally works as follows: it first commits to 2θ strings that are computed starting from the same message, according to Fig. 2. Namely, to commit, first θ random strings are sampled, then another θ strings are computed by XORing the first random strings with the message to commit to. Note that such 2θ strings can be paired such that XORing them again allows one to obtain θ times the same message m, as shown in Step 1a. Then, a commitment is computed for each of the 2θ strings, thus obtaining 2θ commitments.

The above-described procedure is repeated r times, where r is a parameter hardcoded into the algorithm description that depends on λ. Thus, we denote with a_j the set of the 2θ commitments, for $j \in [r]$, and we denote by $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$ the set of all $r(2\theta)$ commitments.

Then, $\forall j \in [r]$, considering the corresponding 2θ commitments (i.e., the corresponding a_j), we randomly sample a θ-bit challenge string $\mathsf{c}_j = (\mathsf{c}_{j,1} \| \ldots \| \mathsf{c}_{j,\theta})$ and compute θ decommitments of the commitments inside a_j according to c_j (i.e., we compute the decommitment of $\tau_{j,1}^{\mathsf{c}_{j,1}}, \ldots, \tau_{j,\theta}^{\mathsf{c}_{j,\theta}}$) and store the values to run $\mathsf{Com.Rec}''$ (i.e., the decommitment algorithm) into z_j, such values for each commitment are the decommitment information (which depends on the underlying commitment scheme Com'') and the message being committed to.

Finally, let msg be the tuple $(A, j, \mathsf{c}_j, \mathsf{z}_j)$, namely all the $r(2\theta)$ commitments, the iteration index, the θ-bit challenge, and the θ decommitment information, the sender interact with $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{msg}))$, and obtains the tuple $(\mathsf{HashConfirm}, h)$. If the output h, truncated to the first ℓ-bits, corresponds to the value 0, the j-th iteration is concluded; otherwise, the computation is repeated, starting with sampling a new c_j. After r completed iterations, the output of the commitment algorithm $\mathsf{Com.Sen}'$ is the set of all r tuples $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$.

For consistency, we also described in Fig. 5 the decommitment algorithm $\mathsf{Com.Rec}'$ that follows the same logic as the one described in Fig. 2. The idea is as follows. During the commitment phase (i.e., the execution of $\mathsf{Com.Sen}'$), the algorithm outputs a commitment $\tau = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ that contains decommitment information for $r\theta$ commitments of Com'' (out of the total $r(2\theta)$ that are present), allowing verification that these commitments can be successfully opened. During the decommitment phase, $\mathsf{Com.Rec}'$ receives decommitment information for the remaining $r\theta$ commitments and verifies that these can be successfully opened. Finally, $\mathsf{Com.Rec}'$ checks that all successfully opened message pairs (half revealed during the commitment phase in τ and half during decommitment) are shares of the same message m, meaning that each pair $(s_{j,i}^0, s_{j,i}^1)$ committed in each a_j satisfies $\mathsf{m} = s_{j,i}^0 \oplus s_{j,i}^1$, with $j \in [r]$ and $i \in [\theta]$.

[15]The core idea in Fischlin's approach is as follows: to generate a valid proof, the prover is required to solve a "proof of work" challenge. Given the initial message a, the prover searches for a transcript $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ such that the RO output, with input the transcript, begins with ℓ zeroes. This transcript then serves as the final proof. A malicious prover who only uses a single transcript $(\mathsf{a}, \mathsf{c}, \mathsf{z})$ will succeed with probability at most $2^{-\ell}$. Therefore, any prover that succeeds with significantly higher probability must have queried the RO on at least two valid transcripts corresponding to different challenges. To achieve soundness error $2^{-\lambda}$, this procedure is repeated r times, where $r = \lambda/\ell$, resulting in a proof size of $B(\lambda/\ell)$, where B denotes the size (in bits) of the original Σ-protocol transcript.

Let $\mathsf{Com}'' = (\mathsf{Com.Sen}'', \mathsf{Com.Rec}'')$ be a non-interactive commitment scheme with message space $\{0,1\}^m$. The security parameter λ defines the parameter m, and the parameters $\ell, r, \theta \in \mathbb{N}$. These integers satisfy $r\ell = \lambda$ and $r > 1$. We describe a non-interactive extractable commitment scheme $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ with message space $\{0,1\}^m$.
Common input: Security parameter λ, the parameters $\ell, r \in \mathbb{N}$, the parameter θ and the message length m and a string $\mathsf{id} \in \{0,1\}^*$.

Commitment phase. On input $\mathsf{m} \in \{0,1\}^m$ and a string $\mathsf{id} \in \{0,1\}^*$, the algorithm $\mathsf{Com.Sen}'$ executes the following steps.

1. For every $j \in [r]$:
   (a) Generate θ random strings $\{s^0_{j,i}\}_{i\in[\theta]}$ of the same length as m, and compute $\{s^1_{j,i} = \mathsf{m} \oplus s^0_{j,i}\}_{i\in[\theta]}$, so that $\{\mathsf{m} = s^0_{j,i} \oplus s^1_{j,i}\}_{i\in[\theta]}$.
   (b) Using Com'', commit to the θ pairs $\{s^0_{j,i}, s^1_{j,i}\}_{i\in[\theta]}$. That is, compute $\{(\tau^0_{j,i}, \mathsf{d}^0_{j,i}) = \mathsf{Com.Sen}''(s^0_{j,i}), (\tau^1_{j,i}, d^1_{j,i}) = \mathsf{Com.Sen}''(s^1_{j,i})\}_{i\in[\theta]}$. Let $\mathsf{a}_j = (\tau^0_{j,1}, \tau^1_{j,1}), \ldots, (\tau^0_{j,\theta}, \tau^1_{j,\theta})$.
   Let $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$.
2. For every $j \in [r]$:
   (a) Set $\mathcal{E}_j = \varnothing$.
   (b) Sample $\mathsf{c}_j \leftarrow\!\!\$\ \{0,1\}^\theta \setminus \mathcal{E}_j$ as the challenge string, where we denote each bit of the challenge string as $\mathsf{c}_j = (\mathsf{c}_{j,1} \| \ldots \| \mathsf{c}_{j,\theta})$.
   (c) Given $\mathsf{a}_j = (\tau^0_{j,1}, \tau^1_{j,1}), \ldots, (\tau^0_{j,\theta}, \tau^1_{j,\theta})$, for each pair decommit to one of the two commitments, that is decommit to $\tau^{\mathsf{c}_{j,1}}_{j,1}, \ldots, \tau^{\mathsf{c}_{j,\theta}}_{j,\theta}$. Specifically store the information necessary to run the $\mathsf{Com.Rec}''$ algorithm in $\mathsf{z}_j = (\mathsf{d}^{\mathsf{c}_{j,1}}_{j,1}, s^{\mathsf{c}_{j,1}}_{j,1}), \ldots, (\mathsf{d}^{\mathsf{c}_{j,\theta}}_{j,\theta}, s^{\mathsf{c}_{j,\theta}}_{j,\theta})$.
   (d) Given msg be the tuple $(A, j, \mathsf{c}_j, \mathsf{z}_j)$, interact with $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{msg}))$, upon receiving the tuple $(\mathsf{HashConfirm}, h)$ check if the first ℓ-bits of h are different from 0, if so set $\mathcal{E}_j \leftarrow \mathcal{E}_j \cup \{\mathsf{c}_j\}$ and go to Step 2b.
3. Output $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$.

The commitment $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ is considered well-formed (i.e., belongs to the domain) if for every $j \in [r]$:

1. Given msg be the tuple $(A, j, \mathsf{c}_j, \mathsf{z}_j)$, the receiver interacts with $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{msg}))$, upon receiving the tuple $(\mathsf{HashConfirm}, h)$ checks if the first ℓ-bits of h are equal to 0, and
2. for every $i \in [\theta]$, $\mathsf{Com.Rec}''$ outputs 1 on input the decommitment value $\mathsf{d}^{\mathsf{c}_{j,i}}_{j,i}$, the commitment $\tau^{\mathsf{c}_{j,i}}_{j,i}$, and the message $s^{\mathsf{c}_{j,i}}_{j,i}$.

Decommitment phase. On input the decommitment information d, the commitment $\tau = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$, and the message to decommit to $\mathsf{m} \in \{0,1\}^m$, the algorithm $\mathsf{Com.Rec}'$ executes the following steps.

1. Parse d as $\bar{\mathsf{z}_1}, \ldots, \bar{\mathsf{z}_r}$ where for every $j \in [r]$, the value $\bar{\mathsf{z}_j} = (\mathsf{d}^{\mathsf{c}_{\bar{j},1}}_{j,1}, s^{\mathsf{c}_{\bar{j},1}}_{j,1}), \ldots, (\mathsf{d}^{\mathsf{c}_{\bar{j},\theta}}_{j,\theta}, s^{\mathsf{c}_{\bar{j},\theta}}_{j,\theta})$, with $\mathsf{c}_{\bar{j},i} = (1 - \mathsf{c}_{j,i})$ for every $i \in [\theta]$.
2. Output 1 if for every $j \in [r]$:
   (a) for every $i \in [\theta]$ the algorithm $\mathsf{Com.Rec}''$ outputs 1 on input the decommitment value $\mathsf{d}^{\mathsf{c}_{\bar{j},i}}_{j,i}$, the commitment $\tau^{\mathsf{c}_{\bar{j},i}}_{j,i}$, and the message $s^{\mathsf{c}_{\bar{j},i}}_{j,i}$, that are values stored in $\bar{\mathsf{z}_j}$ (that is part of d) and a_j (that is part of τ), and
   (b) for every $i \in [\theta]$ the pair $(s^{\mathsf{c}_{j,i}}_{j,i}, s^{\mathsf{c}_{\bar{j},i}}_{j,i})$ satisfy $\mathsf{m} = s^{\mathsf{c}_{j,i}}_{j,i} \oplus s^{\mathsf{c}_{\bar{j},i}}_{j,i}$, that are values in z_j and $\bar{\mathsf{z}_j}$.
3. Otherwise output 0.

Fig. 5: Non-interactive straight-line extractable commitment scheme obtained by transforming the commitment scheme in Fig. 2. Note that $\mathsf{Com.Sen}'$ algorithm has access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality, while $\mathsf{Com.Rec}'$ algorithm does not.

COMMITMENT SIZE. Assuming C is the length in bits of the commitment output by $\mathsf{Com.Sen}''$ and D is the length in bits of the decommitment information output by $\mathsf{Com.Sen}''$, when dealing with a message

of length m. For each tuple $(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)$, the length of a_j is $C(2\theta)$ bits, the length of c_j is θ bits, and the length of z_j is $D\theta$ bits. Thus, an entire transcript has length $(C(2\theta) + \theta + D\theta)$ bits, making the commitment output by $\mathsf{Com.Sen}'$ of total length $r\theta(2C + D + 1)$ bits.

ACCEPTANCE OF WELL-FORMED COMMITMENT. The commitment $\tau = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ is considered well-formed (i.e., belongs to the domain of valid commitments for this scheme) if for each $j \in [r]$: $\mathcal{H}_\ell(A, j, \mathsf{c}_j, \mathsf{z}_j) = 0$ and $\mathsf{Com.Rec}''$ outputs 1 for all θ commitments $(\tau_{j,1}^{\mathsf{c}_{j,1}}, \ldots, \tau_{j,\theta}^{\mathsf{c}_{j,\theta}})$, opened considering $\mathsf{z}_j = (\mathsf{d}_{j,1}^{\mathsf{c}_{j,1}}, s_{j,1}^{\mathsf{c}_{j,1}}), \ldots, (\mathsf{d}_{j,\theta}^{\mathsf{c}_{j,\theta}}, s_{j,\theta}^{\mathsf{c}_{j,\theta}})$ (i.e., all θ commitments are correctly decommitted).

Provability. By inspection of $\mathsf{Com.Rec}'$ in Fig. 5 it follows that the commitment scheme $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ satisfies the provability property of Def. 5. Specifically, $\mathsf{Com.Rec}'$ has no access to the interfaces of the $\mathcal{G}_{\mathsf{oRO}}$ functionality and admits an explicit circuit representation whose size is $\mathsf{poly}(\lambda)$.

4.2 Correctness and Security

We analyze the correctness in App. C, which is straightforward. The intuition behind the proof of security already appears in Sec. 2. We therefore proceed directly to the description of the theorems for hiding and straight-line extractability and we defer their proofs to Apps. D and E. We also discuss in App. E that our commitment scheme suffers from over-extraction, and we emphasize this point here since we later leverage this fact in both the construction and the security analysis of our main result.

Theorem 1 (Hiding). *Consider the commitment scheme* $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ *in Fig. 5. Let* $\mathsf{Com}'' = (\mathsf{Com.Sen}'', \mathsf{Com.Rec}'')$ *be a commitment scheme and* $\mathsf{id} \in \{0,1\}^*$ *a common input. Then, for any* PPT *adversary* $\mathcal{A}$ *playing the game* HDN *against the commitment scheme* Com'*, there exists a* PPT *adversary* $\mathcal{A}_0$ *playing the game* HDN *against* Com''*, such that for every* $\lambda \in \mathbb{N}$, $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}'}(\mathcal{A}, \lambda)$ *reduces to* $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}''}(\mathcal{A}_0, \lambda)$*, where the algorithm* $\mathsf{Com.Sen}'$ *and the adversaries have access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

We prove that the commitment scheme $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ in Fig. 5 is hiding. In a nutshell, proof follows a similar approach to that of Pass and Wee [PW09]. The core idea is to reduce the hiding property of our scheme to the hiding property of the underlying commitment scheme.

Theorem 2 (Straight-Line Extraction). *Let* $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ *be the commitment scheme defined in Fig. 5, and let* $\mathcal{G}_{\mathsf{oRO}}$ *be the functionality described in Fig. 1 and* $\mathsf{id} \in \{0,1\}^*$ *a common input. Then,* Com' *satisfies extractability as in Def. 5. That is, there exists a* PPT *extractor algorithm* $\mathsf{Com.Ext}$ *such that for every commitment* τ*, if a* PPT *adversary* $\mathcal{A}$ *succeeds in decommitting* τ *to some message* m *with non-negligible probability, then* $\mathsf{Com.Ext}(\tau, \mathcal{Q}) = \mathsf{m}$ *with overwhelming probability, where* $\mathcal{Q}$ *is the query-answer list outputted by a call to the* `Observe` *interface of the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

We prove that our commitment scheme Com' from Fig. 5 satisfies the straight-line extractability property according to Def. 5. That is, we exhibit an extractor algorithm that, given as input only τ the commitment outputted by $\mathsf{Com.Sen}'$ (such that τ can be successfully decommitted) and the query-answer list $\mathcal{Q}$ outputted by a call to the `Observe` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality (where the query is done after the adversary output), succeeds in extracting the committed message m except with negligible probability.

We explain how to instantiate such a commitment relying only on 1-to-1 OWFs in App. F.

5 Functionality for UC Blind Signatures

We refer to App. B for the standard notation and preliminaries regarding the (generalized) UC frameworks (GUC) [Can20, CDPW07].

We adopt the functionality for blind signatures introduced by Fischlin [Fis06a], which has become the standard reference functionality in this domain. Nevertheless, there exist (to our knowledge) at least two other functionalities for blind signatures, none of which provide a comparative analysis to guide selection. We make such a comparison of these functionalities and justify our preference for Fischlin's approach in App. G. We describe Fischlin's blind signature functionality $\mathcal{F}_{\mathsf{BS}}$ in Fig. 6. Additional details and an explanation of the functionality are provided in App. H. We slightly deviate from Fischlin's original functionality [Fis06a] by explicitly considering the possibility that the signer aborts, which is only implicit in Fischlin's formulation.

Functionality $\mathcal{F}_{\mathsf{BS}}$

Variables: Initially empty lists $\mathsf{List}_{\mathsf{pk}}$ and List_{σ}.

1. `KeyGen`. On input $(\mathsf{KGQuery}, \mathsf{sid})$ from a party $\mathcal{S}$, check if $\mathsf{sid} = (\mathcal{S}, \mathsf{sid}')$ for some sid'. If not, ignore. Else, forward $(\mathsf{KGQuery}, \mathsf{sid})$ to the adversary. Upon receiving $(\mathsf{VerificationKey}, \mathsf{sid}, \mathsf{pk}, \mathsf{BlSig})$ from the adversary:
   - Output $(\mathsf{VerificationKey}, \mathsf{sid}, \mathsf{pk})$ to $\mathcal{S}$.
   - Record the tuple $(\mathcal{S}, \mathsf{pk}, \mathsf{BlSig})$ in $\mathsf{List}_{\mathsf{pk}}$.
2. `Sign`. On input $(\mathsf{SignQuery}, \mathsf{sid}, \mathsf{m}, \mathsf{pk})$ from some party $\mathcal{U}$, check that $\mathsf{sid} = (\mathcal{S}, \mathsf{sid}')$ for some sid'. If not, ignore. Otherwise, find $(\mathcal{S}, \mathsf{pk}, \mathsf{BlSig})$ such that $(\mathcal{S}, \mathsf{pk}, \mathsf{BlSig}) \in \mathsf{List}_{\mathsf{pk}}$ and:
   - If $\mathcal{U}$ is honest:
     (a) Inform the adversary of the signing request via $(\mathsf{SignDone}, \mathsf{sid})$.
     (b) Inform $\mathcal{S}$ of the signing request via $(\mathsf{SignDone}, \mathsf{sid})$ and receive $(\mathsf{AgreeSignDone}, y)$.
     (c) Set $y' = 1$ then set $y' = y$.
     (d) Compute $\sigma \leftarrow \mathsf{BlSig}(\mathsf{m})$ and if $y' = 1$ send $(\mathsf{Signature}, \mathsf{sid}, \mathsf{m}, \sigma)$ to $\mathcal{U}$.
     (e) Record the tuple $(\mathsf{m}, \sigma, \mathsf{pk}, y')$ in List_{σ}.
   - If $\mathcal{U}$ is corrupt:
     (a) Send $(\mathsf{SignQuery}, \mathsf{sid}, \mathsf{m})$ to the adversary and receive $(\mathsf{Signature}, \mathsf{sid}, \mathsf{m}, \sigma)$.
     (b) Abort if $(\mathsf{m}, \sigma, \mathsf{pk}, 0) \in \mathsf{List}_{\sigma}$.
     (c) Inform $\mathcal{S}$ of the signing request via $(\mathsf{SignDone}, \mathsf{sid})$ and receive $(\mathsf{AgreeSignDone}, y)$.
     (d) Set $y' = 1$ then set $y' = y$.
     (e) Record the tuple $(\mathsf{m}, \sigma, \mathsf{pk}, y')$ in List_{σ}.
3. `Verify`. On input $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$ from a party $\mathcal{P}$, send $(\mathsf{Verify}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$ to the adversary. Upon receiving $(\mathsf{Verification}, \mathsf{sid}, \mathsf{m}, \sigma, b)$ from the adversary:
   - If succeed in find $(\mathsf{m}, \sigma, \mathsf{pk}, 1) \in \mathsf{List}_{\sigma}$ and succeed in find $(\mathcal{S}, \mathsf{pk}, \mathsf{BlSig}) \in \mathsf{List}_{\mathsf{pk}}$, then set $b' = 1$.
   - If not succeed in find $(\mathsf{m}, *, \mathsf{pk}, 1) \in \mathsf{List}_{\sigma}$ and, succeed in find $(\mathcal{S}, \mathsf{pk}, \mathsf{BlSig}) \in \mathsf{List}_{\mathsf{pk}}$ and $\mathcal{S}$ is honest, then set $b' = 0$ and record the entry $(\mathsf{m}, \sigma, \mathsf{pk}, 0)$ in List_{σ}.
   - Else, if $(\mathsf{m}, \sigma, \mathsf{pk}, b'') \in \mathsf{List}_{\sigma}$, then set $b' = b''$.
   - Else, set $b' = b$ and record $(\mathsf{m}, \sigma, \mathsf{pk}, b) \in \mathsf{List}_{\sigma}$.

   Output $(\mathsf{VerifyDone}, \mathsf{sid}, \mathsf{m}, \sigma, b')$ to $\mathcal{P}$.

Fig. 6: The functionality $\mathcal{F}_{\mathsf{BS}}$ from [Fis06a]. We denote the signer by $\mathcal{S}$ and the user by $\mathcal{U}$.

6 GUC Blind Signatures from 1-to-1 OWFs

We present a 2-round protocol, denoted ϕ_{BS}, that GUC-realizes the functionality $\mathcal{F}_{\mathsf{BS}}$ in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model. Our construction relies on the following primitives: a signature scheme $\mathsf{Sig} = (\mathsf{Sig.KG}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$, the non-interactive straight-line extractable and provable commitment scheme $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$, according to Def. 5, that is constructed from any non-interactive commitment $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ as shown in Fig. 5, and two NIWI argument of knowledge schemes $\mathsf{Niwi} = (\mathsf{Niwi.Prove}, \mathsf{Niwi.Verify})$ for NP-relations $\mathsf{R}_{\mathcal{S}}$ and $\mathsf{R}_{\mathcal{U}}$.

Before presenting the relations $\mathsf{R}_{\mathcal{S}}$ and $\mathsf{R}_{\mathcal{U}}$, we briefly describe the key generation procedure (see Fig. 8 for a detailed description). The key generation proceeds as follows: 1) sample random coins $r \leftarrow\!\!\$\ \{0,1\}^*$, 2) run $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{Sig.KG}(1^\lambda; r)$ using r as the algorithm randomness, 3) run the algorithm $\mathsf{Com.Sen}$ on input the message $\mathsf{sk}' \| r$, and obtain the commitment $\tau_{\mathcal{S}}$ and the decommitment information $\mathsf{d}_{\mathcal{S}}$, 4) set the blind signature key pair as $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ and $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$. The NP-relations for the NIWIs are in Fig. 7.

$\mathsf{R}_{\mathcal{S}}(\mathsf{stm} = (\tau_{\mathcal{U}}, \mathsf{pk}), \mathsf{wts} = (\mathsf{m}_0, \mathsf{m}_1, \mathsf{d}_0, \mathsf{d}_1, \mathsf{sk}))$:

1. Unpack $\tau_{\mathcal{U}} = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j \in [r]}$, where $\mathsf{a}_j = \{(\tau^0_{j,i}, \tau^1_{j,i})\}_{i \in [\theta]}$.
2. Set $b' \leftarrow 0$.
3. If $\big(\exists j \in [r], i \in [\theta] : \forall k \in \{0,1\}\ \mathsf{Com.Rec}'(\mathsf{d}_k, \tau^k_{j,i}, \mathsf{m}_k) = 1\big)$ then set $b' \leftarrow 1$.
4. Unpack $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ and $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$
5. Compute $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{Sig.KG}(1^\lambda; r)$
6. $b \leftarrow \big((\mathsf{pk}' = \mathsf{pk}'') \wedge (\mathsf{sk}' = \mathsf{sk}'') \wedge \mathsf{Com.Rec}(\mathsf{d}_{\mathcal{S}}, \tau_{\mathcal{S}}, (\mathsf{sk}' \| r)) = 1\big)$

7. Output $b \vee b'$

$\mathsf{R}_{\mathcal{U}}(\mathsf{stm} = (\mathsf{pk}, \mathsf{m}), \mathsf{wts} = (\mathsf{d}_{\mathcal{U}}, \sigma, \mathsf{sk}, \tau_{\mathcal{U}}))$:

1. Unpack $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ and $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$
2. Compute $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{Sig.KG}(1^{\lambda}; r)$
3. $b' \leftarrow ((\mathsf{pk}' = \mathsf{pk}'') \wedge (\mathsf{sk}' = \mathsf{sk}''))$
4. $b \leftarrow \big(\mathsf{Sig.Verify}(\mathsf{pk}', \tau_{\mathcal{U}}, \sigma) = 1 \wedge \mathsf{Com.Rec}(\mathsf{d}_{\mathcal{U}}, \tau_{\mathcal{U}}, \mathsf{m}) = 1\big)$
5. Output $(b \vee b')$

Fig. 7: The relations $\mathsf{R}_{\mathcal{S}}$ and $\mathsf{R}_{\mathcal{U}}$ used in the protocol ϕ_{BS}.

The relation $\mathsf{R}_{\mathcal{S}}$ is satisfied if either: (a) There exist indices $j \in [r]$ and $i \in [\theta]$ such that the commitments $\tau^0_{j,i}$ and $\tau^1_{j,i}$ part of a pair from a_j (which is part of $\tau_{\mathcal{U}}$) can be successfully opened to messages m_0 and m_1 by leveraging d_0 and d_1, respectively[16]. (b) There exists a secret key $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ such that $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{Sig.KG}(1^{\lambda}; r)$ yields (a) $\mathsf{pk}'' = \mathsf{pk}'$, where $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$ and (b) $\mathsf{sk}'' = \mathsf{sk}'$, where $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$, and $\tau_{\mathcal{S}}$ decommits to $\mathsf{sk}' \| r$ using $\mathsf{d}_{\mathcal{S}}$.

The relation $\mathsf{R}_{\mathcal{U}}$ is satisfied if either: (a) there exists a secret key $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ such that $(\mathsf{sk}'', \mathsf{pk}'') \leftarrow \mathsf{Sig.KG}(1^{\lambda}; r)$ yields (a) $\mathsf{pk}'' = \mathsf{pk}'$, where $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$, and (b) $\mathsf{sk}'' = \mathsf{sk}'$, where $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$; (b) σ is valid under pk' on message $\tau_{\mathcal{U}}$, and $\tau_{\mathcal{U}}$ correctly decommit to m leveraging $\mathsf{d}_{\mathcal{U}}$ as decommitment information.

6.1 Description of the UC Protocol ϕ_{BS}

The protocol ϕ_{BS} is described in Fig. 8, and relizes the $\mathcal{F}_{\mathsf{BS}}$ functionality in Fig. 6.

Protocol ϕ_{BS}

- `KeyGen`: Upon receiving $(\mathsf{KGQuery}, \mathsf{sid})$, the signer $\mathcal{S}$ proceeds as follows:
  1. If $\mathsf{sid} \neq (\mathcal{S}, \mathsf{sid}')$ for some sid', ignore the request.
  2. Otherwise, sample random coins $r \leftarrow\!\!\$\ \{0,1\}^*$.
  3. Run the key generation algorithm $\mathsf{Sig.KG}$ on input 1^{λ} using randomness r to obtain $(\mathsf{sk}', \mathsf{pk}')$.
  4. Run the algorithm $\mathsf{Com.Sen}$ on input sid, the message $(\mathsf{sk}' \| r)$, and obtain the commitment $\tau_{\mathcal{S}}$ and the decommitment information $\mathsf{d}_{\mathcal{S}}$.
  5. Set sk equals to $(\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ and pk equals to $(\mathsf{pk}', \tau_{\mathcal{S}})$. Store sk and output $(\mathsf{VerificationKey}, \mathsf{sid}, \mathsf{pk})$.
- `Sign`: Upon receiving $(\mathsf{SignQuery}, \mathsf{sid}, \mathsf{m}, \mathsf{pk})$, $\mathcal{U}$ proceeds as follows:
  1. If $\mathsf{sid} \neq (\mathcal{S}, \mathsf{sid}')$ for some sid', ignore the request.
  2. Otherwise, initiate a run of the interactive protocol with signer $\mathcal{S}$ where the common inputs of the parties are the security parameter λ and the public key $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$, while the signer $\mathcal{S}$ private input is $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$, and the user $\mathcal{U}$ private input is m. That is, the interaction works as follows:
     (a) **First round of interaction** $(\mathcal{U} \rightarrow \mathcal{S})$**.**
        i. Run the algorithm $\mathsf{Com.Sen}$ on input sid and the message m, and obtain the commitment $\tau_{\mathcal{U}}$ and the decommitment information $\mathsf{d}_{\mathcal{U}}$.
        ii. Send $\tau_{\mathcal{U}}$ to $\mathcal{S}$.
     (b) **Second round of interaction** $(\mathcal{S} \rightarrow \mathcal{U})$**.**
        i. Run the algorithm $\mathsf{Sig.Sign}$ on input the signing key sk' and the message $\tau_{\mathcal{U}}$, and obtain the signature σ'.
        ii. Run the algorithm $\mathsf{Niwi.Prove}$ for the relation $\mathsf{R}_{\mathcal{S}}$ on input sid, the statement $(\tau_{\mathcal{U}}, \mathsf{pk})$ and the witness sk (appropriately padded with dummy values to match the required witness format), and obtain the proof π'.
        iii. Send (σ', π') to $\mathcal{U}$.
     (c) **Signature computation** $(\mathcal{U})$**.**
        i. Abort if: (i) the algorithm $\mathsf{Sig.Verify}$, on input the verification key pk' the commitment $\tau_{\mathcal{U}}$ (as the message) and the signature σ', it outputs 0 or, (ii) if the algorithm

[16] Note that the relation is tight for the straight-line extractable commitment introduced in Sec. 4.1; specifically, we exploit its structure and the fact that it suffers from over-extraction, as discussed above.

Niwi.Verify for the relation $\mathsf{R}_{\mathcal{S}}$, on input the statement $(\tau_{\mathcal{U}}, \mathsf{pk})$ and the proof π', it outputs 0.

ii. Run the algorithm Niwi.Prove for the relation $\mathsf{R}_{\mathcal{U}}$ on input sid, the statement $(\mathsf{pk}, \mathsf{m})$ and the witness $(\mathsf{d}_{\mathcal{U}}, \sigma', 1^{|\mathsf{sk}|}, \tau_{\mathcal{U}})$, and obtain the proof π.

iii. Obtain the message-signature pair (m, σ), where σ is equal to π.

3. Upon completion output $(\texttt{Signature}, \mathsf{sid}, \mathsf{m}, \sigma)$.

– `Verify`: Upon receiving $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$, a party $\mathcal{P}$ proceeds as follows:

1. If $\mathsf{sid} \neq (\mathcal{S}, \mathsf{sid}')$ for some sid', ignore the request.
2. Otherwise, parse σ as π.
3. Compute $b \leftarrow \mathsf{Niwi.Verify}(\mathsf{sid}, \mathsf{stm}, \pi)$ for relation $\mathsf{R}_{\mathcal{S}}$, where stm is equal to $(\mathsf{pk}, \mathsf{m})$.
4. Output $(\mathsf{VerifyDone}, \mathsf{sid}, \mathsf{m}, \sigma, b)$.

Fig. 8: The protocol ϕ_{BS} for the functionality $\mathcal{F}_{\mathsf{BS}}$.

In the following, we state the conditions under which ϕ_{BS} GUC-realizes $\mathcal{F}_{\mathsf{BS}}$ functionality in Fig. 6 in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model.

Theorem 3. *Let* Sig *be a signature scheme;* Com *be the non-interactive, straight-line extractable and provable commitment scheme as in Def. 5, described in Fig. 5;* $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ *and* $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ *be two NIWI argument of knowledge systems; and let* $\mathcal{G}_{\mathsf{oRO}}$ *be the functionality described in Fig. 1. Then the protocol* ϕ_{BS} *GUC-realizes the ideal functionality* $\mathcal{F}_{\mathsf{BS}}$ *in the* $\mathcal{G}_{\mathsf{oRO}}$*-hybrid model.*

The intuition behind the proof of security already appears in Sec. 2. We defer the formal proof, including the simulator description, to App. I.

As a direct consequence of Thm. 3, we obtain the following corollary.

Corollary 1. *Assuming the existence of 1-to-1 OWFs, the protocol* ϕ_{BS} *GUC-realizes the ideal functionality* $\mathcal{F}_{\mathsf{BS}}$ *in the* $\mathcal{G}_{\mathsf{oRO}}$*-hybrid model.*

The protocol ϕ_{BS} is based on three components: (1) a signature scheme Sig, which can be constructed from any OWF (see App. A.3), (2) NIWI schemes Niwi for relations $\mathsf{R}_{\mathcal{S}}$ and $\mathsf{R}_{\mathcal{U}}$, which can be unconditionally achieved in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model (see Sec. 3.1), and (3) the straight-line extractable and provable commitment scheme Com, which we proved can be constructed from 1-to-1 OWFs in Cor. 4. By combining these components, the corollary follows. We discuss in App. K how this construction can be enhanced to yield a practical scheme by assuming CRHFs[17]. In App. J, we discuss how to extend such a result to partially blind signatures.

6.2 Basing the Blind Signature Protocol on OWFs

Our protocol ϕ_{BS}, as presented in the previous section and following from Cor. 1, can be instantiated (in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model) under the assumption of 1-to-1 OWFs. In this section, we show how to relax this requirement to the minimal assumption of standard OWFs. The only component in our construction that requires 1-to-1 OWFs is the commitment scheme Com from Fig. 5. This is because Com makes black-box use of a non-interactive commitment scheme, which cannot be constructed from standard OWFs alone [MP12], but requires at least 1-to-1 OWFs. We make the following simple observation: since we employ the NPRO to compute our non-interactive, straight-line extractable, and provable commitment (i.e., we enable Com.Sen to have access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality), the impossibility result of [MP12] does not apply. Specifically, a trivial construction of a non-interactive commitment scheme in the NPRO model is Naor's commitment, where the first message is replaced by the output of a query to the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality on a common, fixed string agreed upon by both the sender and receiver. Note that we exploit the useful property of Naor's commitment scheme [Nao91] (based on OWFs) that its first message (a uniformly random string) can be securely reused across multiple commitments. Thus, the following two corollaries follow easily.

[17] Notice that the signature scheme constructed from OWFs [Rom90] is restricted to fixed-size messages, where the OWF domain size must match the bit-length of the message being signed. To bypass this limitation, we require CRHFs, which easily avoid it by replacing every non-interactive commitment used (to build both our straight-line extractable commitment and the two used NIWI) with a more efficient non-interactive commitment from CRHFs.

Corollary 2. *Assuming the existence of OWFs, there exists a perfectly binding, non-interactive, straight-line extractable, and provable commitment scheme considering the $\mathcal{G}_{\mathsf{oRO}}$ functionality, as in Def. 5.*

Leveraging such a commitment in our construction ϕ_{BS}, we immediately have the following corollary.

Corollary 3. *Assuming the existence of OWFs, the protocol ϕ_{BS} GUC-realizes the ideal functionality $\mathcal{F}_{\mathsf{BS}}$ in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model.*

Acknowledgments

This project is partially supported by Input Output Research Hub (IORH) of the University of Edinburgh, and by Sunday Group Inc. This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU, and partially supported by the AI-Sec Lab initiative between Sapienza University of Rome and the CYBER 4.0 Competence Center. Pierpaolo Della Monica and Ivan Visconti are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

References

AF96. Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, Berlin, Heidelberg, November 1996. doi:10.1007/BFb0034851.

AHIV23. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: lightweight sublinear arguments without a trusted setup. *DCC*, 91(11):3379–3424, 2023. doi:10.1007/s10623-023-01222-8.

AJOR18. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, and Arnab Roy. Improved (almost) tightly-secure simulation-sound QA-NIZK with applications. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 627–656. Springer, Cham, December 2018. doi:10.1007/978-3-030-03326-2_21.

AKSY22. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53. ACM Press, November 2022. doi:10.1145/3548606.3560650.

AO09. Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 435–450. Springer, Berlin, Heidelberg, December 2009. doi:10.1007/978-3-642-10366-7_26.

App21. Apple. About iCloud Private Relay, December 2021. URL: https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf.

Ara22. Ane Fabo Aranzabal. VPN by Google One. *Google*, December 2022. URL: https://blog.google/products/google-one/common-questions-vpn-google-one.

BBD+23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023. doi:10.1007/978-3-031-38554-4_19.

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 701–732. Springer, Cham, August 2019. doi:10.1007/978-3-030-26954-8_23.

BCC+22. Gustavo Banegas, Kévin Carrier, André Chailloux, Alain Couvreur, Thomas Debris-Alazard, Philippe Gaborit, Pierre Karpman, Johanna Loyer, Ruben Niederhagen, Nicolas Sendrier, Benjamin Smith, and Jean-Pierre Tillich. Wave: A code-based hash-and-sign signature scheme, 2022. NIST PQC Submission. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/wave-spec-web.pdf.

BCC+24. Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. Universally composable SNARKs with transparent setup without programmable random oracle, 2024. In the proceedings of *CRYPTO 2025*. URL: https://eprint.iacr.org/2024/1549.

BCMS20. Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Cham, November 2020. doi:10.1007/978-3-030-64378-2_1.

Beu25. Ward Beullens. *Multivariate Blind Signatures Revisited*, page 230–236. Springer Nature Switzerland, 2025. URL: http://dx.doi.org/10.1007/978-3-031-82852-2_10, doi:10.1007/978-3-031-82852-2_10.

BFH+20. Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020. doi:10.1145/3372297.3417893.

BFM+25. Charles Bouillaguet, Thibauld Feneuil, Jules Maire, Matthieu Rivain, Julia Sauvage, and Damien Vergnaud. Blinding post-quantum hash-and-sign signatures. Cryptology ePrint Archive, Paper 2025/895, 2025. Appear in IEEE S&P 2026. URL: https://eprint.iacr.org/2025/895.

BGSS17. Olivier Blazy, Philippe Gaborit, Julien Schrek, and Nicolas Sendrier. A code-based blind signature. In *2017 IEEE International Symposium on Information Theory (ISIT)*, page 2718–2722. IEEE, June 2017. URL: http://dx.doi.org/10.1109/ISIT.2017.8007023, doi:10.1109/isit.2017.8007023.

BHKR25. Nicholas Brandt, Dennis Hofheinz, Michael Klooß, and Michael Reichle. Tightly-secure blind signatures in pairing-free groups, 2025. In the proceedings of *ASIACRYPT 2025*. URL: https://eprint.iacr.org/2024/2075.

BL13. Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 82–99. Springer, Berlin, Heidelberg, December 2013. doi:10.1007/978-3-642-42045-0_5.

BLNS23. Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 16–29. ACM Press, November 2023. doi:10.1145/3576915.3616613.

Blu81. Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO'81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.

Blu87. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

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, June 2003. doi:10.1007/s00145-002-0120-1.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003. doi:10.1007/3-540-36288-6_3.

Can00. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. URL: https://eprint.iacr.org/2000/067.

Can03. Ran Canetti. Universally composable signatures, certification and authentication. Cryptology ePrint Archive, Report 2003/239, 2003. URL: https://eprint.iacr.org/2003/239.

Can20. Ran Canetti. Universally composable security. *J. ACM*, 67(5), September 2020. doi:10.1145/3402457.

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Pairing-free blind signatures from CDH assumptions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 174–209. Springer, Cham, August 2024. doi:10.1007/978-3-031-68376-3_6.

CC25. Ran Canetti and Megan Chen. Universally composable succinct vector commitments and applications. In *TCC 2025: 22th Theory of Cryptography Conference*. Springer Nature Switzerland, December 2025. doi:10.1007/978-3-032-12290-2_21.

CCLY22. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 533–563. Springer, Cham, August 2022. doi:10.1007/978-3-031-15982-4_18.

CD23. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 423–447. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_15.

CDG+18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Cham, April / May 2018. doi:10.1007/978-3-319-78381-9_11.

CDPW07. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Berlin, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_4.

CG08. Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 345–356. ACM Press, October 2008. doi:10.1145/1455770.1455814.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. doi:10.1145/276698.276741.

CGM21. David Chaum, Christian Grothoff, and Thomas Moser. How to issue a central bank digital currency, 2021. URL: https://www.snb.ch/de/publications/research/working-papers/2021/working_paper_2021_03.

Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. doi:10.1007/978-1-4757-0602-4_18.

Che24. Yilei Chen. Quantum algorithms for lattice problems. Cryptology ePrint Archive, Report 2024/555, 2024. URL: https://eprint.iacr.org/2024/555.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014. doi:10.1145/2660267.2660374.

CKM+23. Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Snowblind: A threshold blind signature in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 710–742. Springer, Cham, August 2023. doi:10.1007/978-3-031-38557-5_23.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Berlin, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_7.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. doi:10.1145/509907.509980.

CLZ04. Shirley Cheung, C. H. Lee, and Yunlei Zhao. Universally composable blind signature functionality. *International Journal of Information Technology & Decision Making*, 03(04):673–684, 2004. doi:10.1142/S0219622004001318.

COS20. Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Cham, May 2020. doi:10.1007/978-3-030-45721-1_27.

CPS+16. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR-composition of sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 112–141. Springer, Berlin, Heidelberg, January 2016. doi:10.1007/978-3-662-49099-0_5.

Dam10. Ivan Damgård. On Σ-protocols, 2010. http://www.cs.au.dk/~ivan/Sigma.pdf.

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

DGS+18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: A privacy-enhancing protocol and browser extension. https://privacypass.github.io/, 2018.

DOT21. Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021. doi:10.1145/3460120.3484595.

dPK22. Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336. Springer, Cham, August 2022. doi:10.1007/978-3-031-15979-4_11.

FHKS16. Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Cham, August / September 2016. doi:10.1007/978-3-319-44618-9_21.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Berlin, Heidelberg, August 2005. doi:10.1007/11535218_10.

Fis06a. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Berlin, Heidelberg, August 2006. doi:10.1007/11818175_4.

Fis06b. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model (full version), 2006. Full version, unpublished. URL: https://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications_1/fischlinblind-sigs2006.pdf.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. doi:10.1007/978-3-319-96881-0_2.

FOO93. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer, Berlin, Heidelberg, December 1993. doi:10.1007/3-540-57220-1_66.

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020. doi:10.1007/978-3-030-45724-2_3.

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990. doi:10.1145/100216.100272.

FS10. Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, Berlin, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_10.

GG14. Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, Berlin, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_27.

Gha17. Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473. Springer, Cham, April 2017. doi:10.1007/978-3-319-70972-7_26.

GKO+23. Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 315–346. Springer, Cham, April 2023. doi:10.1007/978-3-031-30617-4_11.

GKR15. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, 2015. doi:10.1145/2699436.

GLOV12. Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. doi:10.1109/FOCS.2012.47.

GMO16. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. doi:10.1017/CBO9780511546891.

Goo20. Google Chrome Developers. Trust tokens. https://developer.chrome.com/docs/privacy-sandbox/trust-tokens/, 2020.

GRS+11. Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, Berlin, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_36.

HBD+20. Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

HHRS07. Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *48th FOCS*, pages 669–679. IEEE Computer Society Press, October 2007. doi:10.1109/FOCS.2007.27.

HIP+21. Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Private Access Tokens. Internet-Draft draft-private-access-tokens-01, Internet Engineering Task Force, October 2021. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-private-access-tokens/01/.

HKKL07. Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer, Berlin, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_18.

HL25. Javier Herranz and Hugo Louiso. Hash-based blind signatures: First steps. Cryptology ePrint Archive, Paper 2025/2097, 2025. first released 2025-11-14; revised 2026-02-09. URL: https://eprint.iacr.org/2025/2097.

HLM+25. Lucjan Hanzlik, Yi-Fu Lai, Marzio Mula, Eugenio Paracucchi, Daniel Slamanig, and Gang Tang. Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum groups actions, 2025. In the proceedings of *ASIACRYPT 2025*. URL: https://eprint.iacr.org/2025/1100.

HLW23. Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Rai-choo! Evolving blind signatures to the next level. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 753–783. Springer, Cham, April 2023. doi:10.1007/978-3-031-30589-4_26.

HM96. Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 201–215. Springer, Berlin, Heidelberg, August 1996. doi:10.1007/3-540-68697-5_16.

HWG+22. Thomas Hanson, Qian Wang, Santosh Ghosh, Fernando Virdia, Anne Reinders, and Manoj R. Sastry. Optimization for SPHINCS+ using intel secure hash algorithm extensions. Cryptology ePrint Archive, Report 2022/1726, 2022. URL: https://eprint.iacr.org/2022/1726.

JAC+22. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

JLO97. Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164. Springer, Berlin, Heidelberg, August 1997. doi:10.1007/BFb0052233.

JS24. Corentin Jeudy and Olivier Sanders. Improved lattice blind signatures from recycled entropy, 2024. In the proceedings of *CRYPTO 2025*. URL: https://eprint.iacr.org/2024/1289.

Kil88. Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. doi:10.1145/62212.62215.

Kiy20. Susumu Kiyoshima. Round-optimal black-box commit-and-prove with succinct communication. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 533–561. Springer, Cham, August 2020. doi:10.1007/978-3-030-56880-1_19.

KK05. Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005. URL: https://eprint.iacr.org/2005/328.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018. doi:10.1145/3243734.3243805.

KLLQ24. Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI-otter: isogeny-based (partially) blind signatures from the class group action with a twist. *DCC*, 92(11):3587–3643, 2024. doi:10.1007/s10623-024-01441-7.

KLP22. Allen Kim, Xiao Liang, and Omkant Pandey. A new approach to efficient non-malleable zero-knowledge. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 389–418, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. https://eprint.iacr.org/2022/767. doi:10.1007/978-3-031-15985-5_14.

KLR21. Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Cham, December 2021. doi:10.1007/978-3-030-92068-5_16.

KLX22. Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 468–497. Springer, Cham, March 2022. doi:10.1007/978-3-030-97131-1_16.

KNR24. Julia Kastner, Ky Nguyen, and Michael Reichle. Pairing-free blind signatures from standard assumptions in the ROM. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 210–245. Springer, Cham, August 2024. doi:10.1007/978-3-031-68376-3_7.

KOS18. Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 286–313. Springer, Cham, November 2018. doi:10.1007/978-3-030-03807-6_11.

KPG99. Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Berlin, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_15.

KR25. Michael Klooß and Michael Reichle. Blind signatures from proofs of inequality, 2025. In the proceedings of *CRYPTO 2025*. URL: https://eprint.iacr.org/2024/2076.

KRS23. Shuichi Katsumata, Michael Reichle, and Yusuke Sakai. Practical round-optimal blind signatures in the ROM from standard assumptions. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 383–417. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8724-5_12.

KRS25. Dmitry Khovratovich, Ron D. Rothblum, and Lev Soukhanov. How to prove false statements: Practical attacks on fiat-shamir, 2025. In the proceedings of *CRYPTO 2025*. URL: https://eprint.iacr.org/2025/118.

KRW24. Michael Klooß, Michael Reichle, and Benedikt Wagner. Practical blind signatures in pairing-free groups. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 363–395. Springer, Singapore, December 2024. doi:10.1007/978-981-96-0875-1_12.

Ks22. Yashvanth Kondi and abhi shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 279–309. Springer, Cham, December 2022. doi:10.1007/978-3-031-22966-4_10.

KSY11. Jonathan Katz, Dominique Schröder, and Arkady Yerukhimovich. Impossibility of blind signatures from one-way permutations. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 615–629. Springer, Berlin, Heidelberg, March 2011. doi:10.1007/978-3-642-19571-6_37.

KZ07. Aggelos Kiayias and Hong-Sheng Zhou. Equivocal blind signatures and adaptive UC-Security. Cryptology ePrint Archive, Report 2007/132, 2007. URL: https://eprint.iacr.org/2007/132.

KZ08. Aggelos Kiayias and Hong-Sheng Zhou. Equivocal blind signatures and adaptive UC-security. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 340–355. Springer, Berlin, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_19.

Lin03. Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th ACM STOC*, pages 683–692. ACM Press, June 2003. doi:10.1145/780542.780641.

Lin08. Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21(2):200–249, April 2008. doi:10.1007/s00145-007-9015-5.

MP12. Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 701–718. Springer, Berlin, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_41.

Nao91. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. doi:10.1007/BF00196774.

NY89. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. doi:10.1145/73007.73011.

Oka06. Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, Berlin, Heidelberg, March 2006. doi:10.1007/11681878_5.

OO92. Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Berlin, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_27.

Pas03. Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_19.

Pas04. Rafael Pass. *Alternative Variants of Zero-Knowledge Proofs*. Licentiate thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2004. URL: https://kth.diva-portal.org/smash/record.jsf?pid=diva2:9472.

Pas11. Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, June 2011. doi:10.1145/1993636.1993652.

PRS02. Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd FOCS*, pages 366–375. IEEE Computer Society Press, November 2002. doi:10.1109/SFCS.2002.1181961.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. doi:10.1007/s001450010003.

PSM17. Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A practical multivariate blind signature scheme. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 437–454. Springer, Cham, April 2017. doi:10.1007/978-3-319-70972-7_25.

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Berlin, Heidelberg, March 2009. doi:10.1007/978-3-642-00457-5_24.

Rom90. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. doi:10.1145/100216.100269.

Ros04. Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 191–202. Springer, Berlin, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_11.

TZ22. Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Cham, May / June 2022. doi:10.1007/978-3-031-07085-3_27.

YZ06. Moti Yung and Yunlei Zhao. Interactive zero-knowledge with restricted random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 21–40. Springer, Berlin, Heidelberg, March 2006. doi:10.1007/11681878_2.

Zha25. Yifan Zhang. Exact coset sampling for quantum lattice algorithms, 2025. doi:10.48550/ARXIV.2509.12341.

A Deferred Preliminaries

Notation. Given $n \in \mathbb{N}^+$ we write $[n]$ for $\{1, \ldots, n\}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\!\$\ \mathcal{X}$. When $\mathcal{A}$ is an algorithm, we write $y \leftarrow \mathcal{A}(x)$ to denote a run of $\mathcal{A}$ on input x and output y; if $\mathcal{A}$ is randomized, then y is a random variable and $\mathcal{A}(x;\rho)$ denotes a run of $\mathcal{A}$ on input x and random coins $\rho \in \{0,1\}^*$.

We denote by $(\mathcal{A}, \mathcal{B})$ a pair of interactive Turing machines $\mathcal{A}$ and $\mathcal{B}$, and denote by $\langle \mathcal{A}, \mathcal{B} \rangle$ the random variable representing the interaction between two interactive Turing machines $\mathcal{A}$ and $\mathcal{B}$. More precisely, we denote by $\tau = \langle \mathcal{A}(x), \mathcal{B}(y) \rangle$ the interactive execution of $(\mathcal{A}, \mathcal{B})$ invoked with inputs x for $\mathcal{A}$, y for $\mathcal{B}$, and producing τ as the transcript of the execution.

We denote the security parameter by $\lambda \in \mathbb{N}$. A function $\mathsf{negl}(\lambda)$ is negligible in λ (or shortly negligible) if it vanishes faster than the inverse of any polynomial (i.e., for any constant $c > 0$ for sufficiently large λ it holds that $\mathsf{negl}(\lambda) \leq \lambda^{-c}$). A machine is said to be probabilistic polynomial time (PPT), or efficient, if it is randomized and its number of steps is polynomial in the input size.

For a random variable $\mathbf{X}$, we write $\Pr[\mathbf{X} = x]$ for the probability that $\mathbf{X}$ takes a particular value x. A distribution ensemble $\mathbf{X} = \{\mathbf{X}(\lambda)\}_{\lambda \in \mathbb{N}}$ is an infinite sequence of random variables indexed by the security parameter $\lambda \in \mathbb{N}$. Two distribution ensembles $\mathbf{X} = \{\mathbf{X}(\lambda)\}_{\lambda \in \mathbb{N}}$ and $\mathbf{Y} = \{\mathbf{Y}(\lambda)\}_{\lambda \in \mathbb{N}}$ are said to be *computationally indistinguishable*, denoted by $\mathbf{X} \overset{\mathrm{c}}{\approx} \mathbf{Y}$, if for every non-uniform PPT algorithm D there exists a negligible function $\mathsf{negl}(\lambda)$ such that:

$$|\Pr[\mathsf{D}(\mathbf{X}(\lambda)) = 1] - \Pr[\mathsf{D}(\mathbf{Y}(\lambda)) = 1]| \leq \mathsf{negl}(\lambda)$$

When the above equation holds for all (even unbounded) distinguishers D, we say that $\mathbf{X}$ and $\mathbf{Y}$ are statistically close, denoted by $\mathbf{X} \overset{\mathrm{s}}{\approx} \mathbf{Y}$.

A.1 Commitment Schemes

A commitment scheme $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ is a two-phase protocol between two PPT interactive algorithms: a sender $\mathsf{Com.Sen}$ and a receiver $\mathsf{Com.Rec}$. The protocol consists of the following two phases. The *commitment phase* where the sender $\mathsf{Com.Sen}$, given as input a message $\mathsf{m} \in \mathcal{M}$, where $\mathcal{M}$ denotes the message space, interacts with the receiver $\mathsf{Com.Rec}$ to generate a commitment $\tau \in \mathcal{C}$, namely $\tau = \langle \mathsf{Com.Sen}(\mathsf{m}), \mathsf{Com.Rec} \rangle$. The *decommitment phase* where the sender $\mathsf{Com.Sen}$ provides the receiver $\mathsf{Com.Rec}$ with decommitment information $\mathsf{d} \in \mathcal{D}$, allowing $\mathsf{Com.Rec}$ to verify that m corresponds to the commitment τ. Formally, we say that $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ is a commitment scheme if it satisfies the properties of correctness, hiding, and binding. We refer to [Gol01] for the explicit definition of such a standard tool.

We omit the domains $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{D}$ when clear from the context since they are typically determined by the specific commitment scheme and depend on the security parameter λ.

3-ROUND COMMITMENT SCHEME. We now give a definition for a class of commitment schemes where the commitment phase is defined by a 3-round protocol.

Definition 6 (3-Round Commitment Scheme). *A 3-round commitment scheme* $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$ *is a two-phase protocol between two* PPT *interactive algorithms according to App. A.1, with the additional constraint that the* commitment phase *consists of exactly 3 rounds of interaction*[18]*. That is, the sender algorithm* $\mathsf{Com.Sen}$ *in a 3-round commitment scheme is composed of two sub-algorithms, denoted by* $\mathsf{Com.Sen} = (\mathsf{Com.Sen}_1, \mathsf{Com.Sen}_2)$ *(for the commitment phase). Given* $\mathsf{m} \in \mathcal{M}$ *be the sender's input , the interaction transcript* τ *is defined as follows:*

$$(\mathsf{a}, \mathsf{st}) \leftarrow \mathsf{Com.Sen}_1(\mathsf{m}); \mathsf{c} \leftarrow \mathsf{Com.Rec}(\mathsf{a}); (\mathsf{d}, \mathsf{z}) \leftarrow \mathsf{Com.Sen}_2(\mathsf{st}, \mathsf{c}).$$

That is, for a 3-round commitment scheme, we have that $\tau = (\mathsf{a}, \mathsf{c}, \mathsf{z})$.

[18] In this work, we consider only non-interactive decommitment phases. Therefore, when we refer to the number of rounds or the round-complexity of a commitment, we refer only to the commitment phase. Note that there is no loss of generality in assuming that the decommitment stage is non-interactive. This is because any such interactive algorithm can be replaced with a non-interactive one as follows: let α be the internal state of the sender when the decommitment starts, and let $\mathsf{Com.Rec}(\alpha)$ simulate the sender and the interactive verifier in the interactive decommitment stage.

Here, the sender first computes an initial message a and an internal (private) state st. The receiver then responds with a challenge c, which the sender uses to compute the final message z. For the *decommitment phase*, since we focus in this paper only on commitment schemes with non-interactive decommitment phase, we implicitly always have that there exists a receiver algorithm $\mathsf{Com.Rec}$ which verifies that the message m sent by $\mathsf{Com.Sen}$ along with the decommitment information d are consistent with $\tau = (\mathsf{a}, \mathsf{c}, \mathsf{z})$ (i.e., that the commitment transcript τ can be decommitted to m leveraging d).

Non-interactive commitment scheme. A non-interactive commitment scheme is defined as a pair of PPT algorithms $\mathsf{Com} = (\mathsf{Com.Sen}, \mathsf{Com.Rec})$, where: a) The sender $\mathsf{Com.Sen}$ takes as input $(\mathsf{m}; \rho)$, where $\mathsf{m} \in \mathcal{M}$ is the message to be committed, and ρ represents the random coins used as input to the algorithm. The algorithm outputs a tuple (τ, d), consisting of the commitment τ (sent to the receiver) and the decommitment information d; b) The receiver $\mathsf{Com.Rec}$ takes as input the tuple $(\mathsf{d}, \tau, \mathsf{m})$ and outputs 1 if, using d, it accepts m as the decommitment of τ, 0 otherwise. We refer to [Gol01] for the explicit definition.

A.2 Non-Interactive Straight-line Extractable Commitment From [Pas04]

It is straightforward to construct efficient commitment schemes that are straight-line extractable in the NPRO model according to Def. 4. Specifically, the $\mathsf{Com.Sen}$ algorithm, on input an identifier id, a message m, and random coins ρ, sends a query to the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{m} \| \rho))$. After receiving the response $(\mathsf{HashConfirm}, h)$, it stores $\mathsf{d} = \rho$ as the decommitment information and outputs the commitment $\tau = h$. In the decommitment phase, the sender reveals the pair (m, d), and the $\mathsf{Com.Rec}$ algorithm, on input the same identifier id, the message m, and the decommitment information d, sends a query to the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{m} \| \mathsf{d}))$. After receiving the response $(\mathsf{HashConfirm}, h')$, it checks if h' equals τ; if so, it outputs 1, otherwise 0.

A proof that this scheme satisfies Def. 4 appears in [Pas04, Lemma 7]. Intuitively, the extractor proceeds by checking the list of queries obtained from the `Observe` interface to the $\mathcal{G}_{\mathsf{oRO}}$ functionality and searching for an input whose output matches the commitment τ. If such a query is found, the extractor outputs (part of) the corresponding answer as the committed message; otherwise, it returns $\perp$. Since a successful opening with non-negligible probability requires querying the RO on the committed value, the extractor succeeds with overwhelming probability.

Note that, informally, a malicious receiver cannot leverage the capabilities of an extractor to break the hiding property of a challenger commitment according to Def. 3. This is because, although such a malicious receiver can query the $\mathcal{G}_{\mathsf{oRO}}$ functionality through the `Observe` interface to obtain the query-answer list made to the $\mathcal{G}_{\mathsf{oRO}}$ functionality, the set of queries it can (additionally) see are only those made with a different id (i.e., the illegitimate queries), according to the idea explained in Sec. 2. Thus, this set does not contain the query and answer that would enable recovery of the committed message. That is, an adversary playing against an honest sender cannot break the hiding property even though it has access to the illegitimate queries through the $\mathcal{G}_{\mathsf{oRO}}$ functionality.

A similar idea applies in the case of a malicious sender who cannot leverage commitments computed with other ids to break the extractability property. In a nutshell, during the decommitment phase, $\mathsf{Com.Rec}$ runs with input a specific id. Since this id is included in the input to the $\mathcal{G}_{\mathsf{oRO}}$ functionality's `Query` interface as part of the message, although the queries might be made for the same base message but with different ids, the effective message is different. According to the description of the $\mathcal{G}_{\mathsf{oRO}}$ functionality, this results in different outputs from the `Query` interface. Thus, if extraction does not succeed because there is no corresponding query, then the decommitment also cannot succeed (except with the negligible probability that a malicious sender correctly guesses the output of the `Query` interface for a given message). That is, an adversary playing against an honest receiver cannot break the straight-line extractability property even though it makes use of queries made to the $\mathcal{G}_{\mathsf{oRO}}$ functionality with another id (i.e., in other sessions).

A.3 Signature Schemes

A signature scheme is a tuple of efficient algorithms $\mathsf{Sig} = (\mathsf{Sig.KG}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ where:

- $\mathsf{Sig.KG}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$ on input the security parameter, outputs a secret (signing) key sk and a public (verification) key pk.

– $\mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{m}) \to \sigma$ on input a signing key sk and a message $\mathsf{m} \in \{0,1\}^*$, outputs a signature σ.
– $\mathsf{Sig.Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \to 0/1$ is deterministic and on input a verification key pk, a message m and a signature σ, outputs 1 if σ is valid and 0 otherwise.

and such that correctness and unforgeability as defined below hold.

Definition 7 (Correctness). *A signature scheme* $\mathsf{Sig} = (\mathsf{Sig.KG}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ *is* perfectly correct *if for every* $\lambda \in \mathbb{N}$ *and every message* $\mathsf{m} \in \{0,1\}^*$*, with probability one over the sampling of the key pair* $(\mathsf{sk}, \mathsf{pk})$*, the signing algorithm* $\mathsf{Sig.Sign}(\mathsf{sk}, \mathsf{m})$ *returns a signature* σ *such that* $\mathsf{Sig.Verify}(\mathsf{pk}, \mathsf{m}, \sigma) = 1$.

Definition 8 (Unforgeability). *Consider the game* EUF-CMA*, described in Fig. 9. A signature scheme* $\mathsf{Sig} = (\mathsf{Sig.KG}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ *satisfies existential unforgeability under chosen-message attacks (*EUF-CMA*) if for every* PPT *adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig}}(\mathcal{A}, \lambda)$ *is negligible in* λ *where* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sig}}(\mathcal{A}, \lambda) := \Pr\left[\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{Sig}}(\lambda) = 1\right]$.

Game $\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{Sig}}(\lambda)$

1. The key generation algorithm $\mathsf{Sig.KG}$ is run on input the security parameter, outputting a key pair $(\mathsf{sk}, \mathsf{pk})$. A set $\mathcal{Q}$ is initialized to $\varnothing$.
2. A signing oracle Sign is initialized with access to the signing key sk and defined as follows:
   On input a message m, the oracle runs the signing algorithm $\mathsf{Sig.Sign}$ on input $(\mathsf{sk}, \mathsf{m})$. It obtains a signature σ, adds m to the set $\mathcal{Q}$, and returns σ.
3. The adversary $\mathcal{A}$ is given pk and access to the oracle Sign. After interacting with the oracle, it outputs a message-signature pair (m^*, σ^*).
4. The game outputs 1 if: **(a)** $\mathsf{m}^* \notin \mathcal{Q}$, and **(b)** $\mathsf{Sig.Verify}(\mathsf{pk}, \mathsf{m}^*, \sigma^*) = 1$. Otherwise 0.

Fig. 9: The EUF-CMA game.

We remark that full-fledged signature schemes handling messages of fixed polynomial size in λ can be constructed under the minimal assumption that OWFs exist[19], that is a classical result due to Rompel [Rom90, KK05]. This builds upon Naor and Yung's work [NY89], who defined universal one-way hash functions (UOWHFs) and showed they imply secure signature schemes, with Rompel later proving that UOWHFs could be constructed from any OWF.

A.4 NIWI from the Existence of the NPRO Model

We show that it is possible to construct a NIWI for NP languages, assuming that the algorithms have only black-box access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality (i.e., without leveraging any hardness assumption). Although this result already appears implicitly in the literature, our goal here is to make the construction explicit by appropriately combining existing components.

Our starting point is Blum's classical protocol for Hamiltonicity [Blu87]. For the commitment scheme in the first round, we rely on the non-interactive straight-line extractable commitment from [Pas04], which we recalled above. We observe two key facts: 1) Blum's protocol is an example of a Σ-protocol; and 2) the 1-bit challenge version of Blum's protocol is ZK (with a constant soundness error). Since ZK implies WI [FS90], it is also WI.

WI is preserved under parallel composition [FS90]. Consequently, by executing the 1-bit challenge version of Blum's protocol in parallel, the soundness error can be reduced to negligible, while maintaining WI. Thus, Blum's protocol results in a Σ-protocol for Hamiltonicity that is also WI. It is worth noting, however, that not all Σ-protocols are WI (see [CPS+16, Appendix A]). Given this, we apply the Fiat-Shamir (FS) transform, by making use of the $\mathcal{G}_{\mathsf{oRO}}$ functionality, to Blum's WI Σ-protocol. According to [YZ06, Claim 1], if the original 3-round public-coin HVZK is also WI, the transformed protocol via FS remains WI. Notably, the proof of WI does not rely on the black-box access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality. This establishes that NIWI for NP can indeed be constructed under the assumption of the existence of a NPRO (i.e., under the assumption that the algorithm has only access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality and no hardness assumption holds).

We highlight that this construction is also an argument of knowledge according to Def. 11, specifically because the commitments in the first round are straight-line extractable by making use of the $\mathcal{G}_{\mathsf{oRO}}$ functionality.

[19] It is worth noting that these signature constructions require the underlying OWF to have a domain of size that matches the size of the messages to be signed. In practice, this means the scheme works for messages of fixed dimensions.

A.5 Non-Interactive Witness Indistinguishability

Given a relation $\mathsf{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, which is a binary relation that runs in polynomial time; for a statement stm, we call wts a witness if $\mathsf{R}(\mathsf{stm}, \mathsf{wts}) = 1$, and define the language $\mathcal{L}_{\mathsf{R}} := \{\mathsf{stm} \mid \exists \mathsf{wts} : \mathsf{R}(\mathsf{stm}, \mathsf{wts}) = 1\}$.

That is, we give a definition here, for a relation R, of a non-interactive witness indistinguishability (NIWI) argument of knowledge system in the NPRO model, where the extraction is done straight-line.

A NIWI argument of knowledge for a relation R is a tuple of efficient PPT algorithms $\mathsf{Niwi}[\mathsf{R}] = (\mathsf{Niwi.Prove}, \mathsf{Niwi.Verify})$ where both algorithms have black-box access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality such that:

- $\mathsf{Niwi.Prove}(\mathsf{id}, \mathsf{stm}, \mathsf{wts}) \to \pi$ on input a statement stm and a witness wts, outputs a proof π.
- $\mathsf{Niwi.Verify}(\mathsf{id}, \mathsf{stm}, \pi) \to 0/1$ is deterministic and on input a statement stm and a proof π, outputs 1 if π is valid and 0 otherwise.

and correctness, witness indistinguishability (WI) and argument of knowledge as defined below hold.

Definition 9 (Correctness). *A NIWI argument of knowledge system* $\mathsf{Niwi}[\mathsf{R}] = (\mathsf{Niwi.Prove}, \mathsf{Niwi.Verify})$*, where both algorithms have black-box access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality, is* perfectly correct *if, for every* $\mathsf{id}, \mathsf{id}' \in \{0,1\}^*$ *and for every pair* $(\mathsf{stm}, \mathsf{wts})$*, such that* $\mathsf{R}(\mathsf{stm}, \mathsf{wts}) = 1$ *the following holds:*

$$\Pr\left[\mathsf{Niwi.Verify}(\mathsf{id}', \mathsf{wts}, \pi) = 1 \wedge \mathsf{id} = \mathsf{id}' \;\middle|\; \pi \leftarrow \mathsf{Niwi.Prove}(\mathsf{id}, \mathsf{stm}, \mathsf{wts})\right] = 1$$

Definition 10 (Witness Indistinguishability). *Consider the game* WI*, described in Fig. 10, parameterized by the security parameter* λ *and a string* $\mathsf{id} \in \{0,1\}^*$*. A NIWI argument of knowledge system* $\mathsf{Niwi}[\mathsf{R}] = (\mathsf{Niwi.Prove}, \mathsf{Niwi.Verify})$ *satisfies* computational WI *if for every* PPT *adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathrm{wi}}_{\mathsf{Niwi}[\mathsf{R}]}(\mathcal{A}, \lambda)$ *is negligible in* λ *where* $\mathsf{Adv}^{\mathrm{wi}}_{\mathsf{Niwi}[\mathsf{R}]}(\mathcal{A}, \lambda) := \left|1/2 - \Pr\left[\mathrm{WI}^{\mathcal{A}}_{\mathsf{Niwi}[\mathsf{R}]}(\lambda, \mathsf{id}) = 1\right]\right|$*, and the algorithms and* $\mathcal{A}$ *have black-box access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

Game $\mathrm{WI}^{\mathcal{A}}_{\mathsf{Niwi}[\mathsf{R}]}(\lambda, \mathsf{id})$

1. A challenge bit is sampled uniformly at random, $b \leftarrow\!\!\$\ \{0,1\}$.
2. A proving oracle Prove is defined as follows:
   On input a string id, a statement stm and two witnesses $\mathsf{wts}_0, \mathsf{wts}_1$, the oracle first checks if both witnesses are valid by verifying that $\mathsf{R}(\mathsf{stm}, \mathsf{wts}_0) = 1$ and $\mathsf{R}(\mathsf{stm}, \mathsf{wts}_1) = 1$. If either witness is invalid, it returns $\perp$. Otherwise, it runs the proving algorithm $\mathsf{Niwi.Prove}(\mathsf{id}, \mathsf{stm}, \mathsf{wts}_b)$ using the challenge witness wts_b, obtains a proof π, and returns it.
3. The adversary $\mathcal{A}$ has as input the security parameter 1^λ, the string id and access to the oracle Prove. After interacting with the oracle, it outputs a bit b'. The game outputs 1 if b is equal to b', otherwise 0.

Fig. 10: The WI game. Note that $\mathcal{A}$ having access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality can interact with the `Observe` interface.

Loosely speaking, WI [FS90] means that an adversary cannot distinguish which one of (at least) two possible witnesses, wts_0 or wts_1, was used in generating the proof. Clearly, such a definition makes sense only when there exists more than one witness for each statement; otherwise, the property is trivially satisfied, without giving any useful guarantees.

Definition 11 (Argument of Knowledge [Pas04]). *Consider the game* AOK*, described in Fig. 11 parameterized by the security parameter* λ *and a string* $\mathsf{id} \in \{0,1\}^*$*. A NIWI argument of knowledge system* $\mathsf{Niwi}[\mathsf{R}] = (\mathsf{Niwi.Prove}, \mathsf{Niwi.Verify})$ *satisfies the argument of knowledge property if a) there exists a* PPT *extractor machine* $\mathsf{Niwi.Ext}$ *and b) for every* PPT *adversary* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathrm{aok}}_{\mathsf{Niwi}[\mathsf{R}]}(\mathcal{A}, \lambda)$ *is negligible in* λ *where* $\mathsf{Adv}^{\mathrm{aok}}_{\mathsf{Niwi}[\mathsf{R}]}(\mathcal{A}, \lambda) := \Pr\left[\mathrm{AOK}^{\mathcal{A}}_{\mathsf{Niwi}[\mathsf{R}]}(\lambda, \mathsf{id}) = 1\right]$*, and the algorithms and* $\mathcal{A}$ *have black-box access to the* $\mathcal{G}_{\mathsf{oRO}}$ *functionality.*

Game $\mathrm{AOK}^{\mathcal{A}}_{\mathsf{Niwi}[\mathsf{R}]}(\lambda)$

1. The adversary $\mathcal{A}$ on input the security parameter 1^λ and the string id, outputs a pair (stm, π).
2. The extractor $\mathsf{Niwi.Ext}$ on input $(\mathsf{id}, \mathcal{Q}, \mathsf{stm}, \pi)$, where $\mathcal{Q}$ is the query-answer list output by a call to the `Observe` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality, outputs a value wts.

3. The game outputs 1 if: **(a)** $\mathsf{Niwi.Verify}(\mathsf{stm}, \pi) = 1$, and **(b)** $\mathsf{R}(\mathsf{stm}, \mathsf{wts}) \neq 1$. Otherwise, the game outputs 0.

Fig. 11: The AOK game. Note that $\mathcal{A}$ having access to the $\mathcal{G}_{\mathsf{oRO}}$ functionality can interact with the `Observe` interface.

Definition for preventing malleability attacks. Following the motivation in the definition of the straight-line extractable commitment, we highlight that we have carefully developed this definition to explicitly manage queries to the $\mathcal{G}_{\mathsf{oRO}}$ global functionality and introduced id because we anticipate that by properly leveraging these identifiers in accordance with queries to the $\mathcal{G}_{\mathsf{oRO}}$ global functionality, it is possible to construct schemes that safely enable the WI property in certain sessions while correctly extracting in other sessions. In App. A.4 we discuss how to instantiate such a NIWI assuming only the existence of the NPRO.

B The Basic and Generalized UC Frameworks

Basic UC. The universal composability (UC) framework [Can00, Can20] is a framework to define and prove the security of protocols. It follows the simulation-based security paradigm, meaning that security of a protocol is defined as the simulatability of the protocol based on an ideal functionality $\mathcal{F}$. In an imaginary ideal world, parties hand their protocol inputs to a trusted party running $\mathcal{F}$, where $\mathcal{F}$ by construction executes the task at hand in a secure manner. A protocol ϕ is considered a secure realization of $\mathcal{F}$ if the real world, in which parties execute the real protocol, is indistinguishable from the ideal world. Namely, for every real-world adversary $\mathcal{A}$ attacking the protocol, we can design an ideal-world attacker (simulator) Sim that performs an equivalent attack in the ideal world. As the ideal world is secure by construction, this means that there are no meaningful attacks on the real-world protocol either.

One of the goals of UC is to simplify the security analysis of protocols by guaranteeing secure composition of protocols and, consequently, allowing for modular security proofs. One can design a protocol ϕ assuming the availability of an ideal functionality $\mathcal{F}'$, i.e., ϕ is a $\mathcal{F}'$-hybrid protocol. If ϕ securely realizes $\mathcal{F}$, and another protocol ϕ' securely realizes $\mathcal{F}'$, then the composition theorem guarantees that ϕ composed with ϕ' (i.e., replacing ϕ' with $\mathcal{F}'$) is a secure realization of $\mathcal{F}$.

Security is defined through an interactive Turing machine (ITM)[20] $\mathcal{Z}$ that models the environment of the protocol and chooses protocol inputs to all participants. Let $\mathrm{EXEC}_{\phi,\mathcal{A},\mathcal{Z}}$ denote the output of $\mathcal{Z}$ in the real world, running with protocol ϕ and adversary $\mathcal{A}$, and let $\mathrm{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ denote its output in the ideal world, running with functionality $\mathcal{F}$ and simulator Sim. Protocol ϕ securely realizes $\mathcal{F}$ if for every PPT adversary $\mathcal{A}$, there exists a simulator Sim such that for every environment $\mathcal{Z}$ the following holds: $\mathrm{EXEC}_{\phi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$.

Generalized UC. A Basic UC protocol in the RO model is modeled as an $\mathcal{F}_{\mathsf{RO}}$-hybrid protocol. Since an instance of a Basic UC functionality can be used only by a single protocol instance, it is necessary to assume that each protocol instance is equipped with its own RO, independent of the ROs used by other instances. As the RO model is supposed to be an idealization of real-world hash functions, this is not a very realistic model. That is, given that we only have a handful of standardized hash functions, it's hard to argue their independence across many protocol instances.

To address these limitations of Basic UC, Canetti et al. [CDPW07] introduced the Generalized UC (GUC) framework, which allows for shared "global" ideal functionalities (denoted by $\mathcal{G}$) that can be used by all protocol instances. Additionally, GUC gives the environment more powers in the UC experiment. Let $\mathrm{GEXEC}_{\phi,\mathcal{A},\mathcal{Z}}$ be defined as $\mathrm{EXEC}_{\phi,\mathcal{A},\mathcal{Z}}$, except that the environment $\mathcal{Z}$ is no longer constrained, meaning that it is allowed to start arbitrary protocols in addition to the challenge protocol ϕ. Similarly, $\mathrm{GIDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ is equivalent to $\mathrm{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$ but $\mathcal{Z}$ is now unconstrained. If ϕ is a $\mathcal{G}$-hybrid protocol, where $\mathcal{G}$ is some shared functionality, then $\mathcal{Z}$ can start additional $\mathcal{G}$-hybrid protocols, possibly learning information about or influencing the state of $\mathcal{G}$.

Definition 12 ([CDPW07]). *A protocol ϕ GUC-realizes ideal functionality $\mathcal{F}$, if for any* PPT *adversary $\mathcal{A}$ there exists a* PPT *simulator* Sim *such that for any unconstrained environment $\mathcal{Z}$ the following holds:* $\mathrm{GEXEC}_{\phi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{GIDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$.

[20] An ITM [GMR89, Section 2.1] is a standard Turing machine extended with special incoming and outgoing communication tapes and a random tape. This enables multiple ITMs to exchange messages, modeling protocol participants interacting with each other.

GUC gives very strong security guarantees, as the unconstrained environment can run arbitrary protocols in parallel with the challenge protocol, where the different protocol instances might share access to global functionalities. However, exactly this flexibility makes it hard to prove the GUC security of protocols. To address this, Canetti et al. [CDPW07] also introduced Externalized UC (EUC). Typically, a protocol ϕ uses many local hybrid functionalities $\mathcal{F}$ but only uses a single shared functionality $\mathcal{G}$. Such protocols are called $\mathcal{G}$-subroutine respecting, and EUC allows for simpler security proofs for such protocols. Rather than considering unconstrained environments, EUC considers $\mathcal{G}$-externally constrained environments. Such environments can invoke only a single instance of the challenge protocol but can additionally query the shared functionality $\mathcal{G}$ through dummy parties that are not part of the challenge protocol. The EUC experiment is equivalent to the Basic UC experiment, except that it considers $\mathcal{G}$-externally constrained environments.

Definition 13 ([CDPW07]). *A $\mathcal{G}$-subroutine respecting protocol ϕ EUC-realizes ideal functionality $\mathcal{F}$, if for any* PPT *adversary $\mathcal{A}$ there exists a* PPT *simulator* Sim *such that for any $\mathcal{G}$-externalized environment $\mathcal{Z}$ the following holds:* $\text{EXEC}^{\mathcal{G}}_{\phi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \text{IDEAL}^{\mathcal{G}}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}$.

In [CDPW07] Canetti et al. showed that for any $\mathcal{G}$-subroutine respecting protocol ϕ, proving that ϕ EUC-realizes $\mathcal{F}$ is equivalent to proving that ϕ GUC-realizes $\mathcal{F}$. Therefore, when we want to prove the GUC security of a protocol, we always turn to EUC security. That is, proving the security of such protocols can be done by considering this (simpler) static setup, rather than having to consider arbitrary protocols running alongside the challenge protocol.

C Correctness of Com' in Fig. 5

The correctness of $\mathsf{Com}' = (\mathsf{Com.Sen}', \mathsf{Com.Rec}')$ follows from the correctness of the underlying scheme $\mathsf{Com}'' = (\mathsf{Com.Sen}'', \mathsf{Com.Rec}'')$. The analysis follows the very same idea of the one in [Ks22, Theorem 6.4]. We denote by Q_{j,c_j} the query made by $\mathsf{Com.Sen}'$, in Step 2d of its algorithm, to the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality. We denote by $\mathsf{E}_{\mathsf{fail}}$ the following event: $\mathsf{E}_{\mathsf{fail}} :\iff \exists j \in [r] : \forall \mathsf{c}_j \in \{0,1\}^\theta, h \neq 0^\ell$. Where we denote by h the output of the $\mathcal{G}_{\mathsf{oRO}}$ functionality to the `Query` interface, according to Fig. 1. That is, the only event in which the sender does not succeed in computing a correct commitment.

Given that, $r\ell = \lambda$ (with $r > 1$) and that each output (with a different input) of the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality is indipendent, we can bound the probability that $\mathsf{E}_{\mathsf{fail}}$ as follows:

$$\begin{aligned}
\Pr[\mathsf{E}_{\mathsf{fail}} = 1] &= \Pr\left[\exists j \in [r] : \forall \mathsf{c}_j \in \{0,1\}^\theta,\, h \neq 0^\ell\right] \\
&\leq \sum_{j\in[r]} \Pr\left[\forall \mathsf{c}_j \in \{0,1\}^\theta,\, h \neq 0^\ell\right] = \sum_{j\in[r]} \prod_{\mathsf{c}_j \in \{0,1\}^\theta} \Pr\left[h \neq 0^\ell\right] \\
&= \sum_{j\in[r]} \prod_{\mathsf{c}_j \in \{0,1\}^\theta} \left(1 - \frac{1}{2^\ell}\right) = r\left(1 - \frac{1}{2^\ell}\right)^{2^\theta} \approx re^{-2^\theta \cdot 2^{-\ell}} = re^{-2^{(\theta-\ell)}}
\end{aligned}$$

On the choice of θ. We now discuss the explicit choice of the parameter $\theta \in \mathbb{N}^+$ for our commitment scheme. First, we note that for the probability of event $\mathsf{E}_{\mathsf{fail}}$ to be negligible, we require that $re^{-2^{(\theta-\ell)}}$ be negligible. This is achieved by ensuring that $2^{(\theta-\ell)}$ is at least equal to the security parameter λ. Thus, we set $\theta = \lceil \log \lambda \rceil + \ell$, which yields approximately $e^{-\lambda}$, clearly negligible in the security parameter.

D Proof of Thm. 1

We prove that our commitment scheme Com' from Fig. 5 satisfies the hiding property according to Def. 3. The proof proceeds via a hybrid argument, reducing to the security of the underlying commitment scheme Com''.

Game_0. This is the hiding game HDN from Fig. 3, instantiated with the commitment scheme Com' from Fig. 5. The adversary $\mathcal{A}$, after choosing two messages $(\mathsf{m}_0, \mathsf{m}_1)$, receives a commitment to message m_b, consisting of tuples $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$. For each $j \in [r]$, a_j contains θ pairs of commitments computed using $\mathsf{Com.Sen}''$ (i..e, $\mathsf{a}_j = (\tau^0_{j,1}, \tau^1_{j,1}), \ldots, (\tau^0_{j,\theta}, \tau^1_{j,\theta})$); c_j is a uniformly random θ-bit string; and z_j contains decommitment information for opening one commitment from each pair according to c_j. The game outputs 1 if $\mathcal{A}$'s guess equals b; otherwise 0.

$\underline{\mathsf{Game}_1}$. Identical to Game_0, except that in the first commitment pair $(\tau^0_{1,1}, \tau^1_{1,1})$ of a_1, we use shares of m_{1-b} instead of m_b. Specifically, when $j = 1$ and $i = 1$, after sampling $s^0_{1,1} \in \{0,1\}^m$ uniformly, we compute $s^1_{1,1}$ as $\mathsf{m}_{1-b} \oplus s^0_{1,1}$. All remaining (i.e., $(\theta - 1) + r(\theta - 1) = r\theta - 1$) commitment pairs still use shares of m_b.

$\underline{\mathsf{Game}_k}$. We generalize the idea behind Game_1. For each $k \in \{2, \ldots, r\theta - 1\}$, Game_k is identical to Game_{k-1} except for one change: in Game_{k-1}, the first $k-1$ commitment pairs[21] are generated using shares of message m_{1-b}, while the remaining $r\theta - (k-1)$ pairs use shares of m_b. In Game_k, the k-th pair is also computed using shares of m_{1-b}.

$\underline{\mathsf{Game}_{r\theta}}$. In this final game, all $r\theta$ commitment pairs are computed using shares of m_{1-b}. Thus, the adversary receives a commitment to m_{1-b} instead of m_b. Note that this game is identical to Game_0 (and thus, to the real game HDN) except that after sampling the bit b, all computations are performed using bit $(1-b)$ instead.

REDUCTION TO THE HIDING PROPERTY OF THE UNDERLYING SCHEME. We need to show that the advantage of the adversary $\mathcal{A}$ in breaking the hiding of the commitment scheme Com' can be reduced to the advantage of breaking the hiding of the underlying commitment Com''.

That is, $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}'}(\mathcal{A}, \lambda) := \left|1/2 - \Pr\left[\mathrm{HDN}^{\mathcal{A}}_{\mathsf{Com}'}(\lambda) = 1\right]\right|$ that is equal to $\left|1/2 - \Pr\left[\mathsf{Game}^{\mathcal{A}}_0(\lambda) = 1\right]\right|$. Observe that in $\mathsf{Game}_{r\theta}$, the output is 1 when the adversary outputs bit b while the challenger uses bit $(1-b)$. We can therefore express this advantage as $\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}'}(\mathcal{A}, \lambda) = 1/2 \left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_0(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{r\theta}(\lambda) = 1\right]\right|$.

We thus need to show that $\left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_0(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{r\theta}(\lambda) = 1\right]\right|$ is bounded by the advantage of an adversary in breaking the hiding of Com''. By a standard hybrid argument, it suffices to prove that games Game_k and Game_{k-1} are computationally indistinguishable, assuming the hiding property of Com''.

$$\begin{aligned} &\left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_0(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{r\theta}(\lambda) = 1\right]\right| \leq \\ &\quad \textstyle\sum_{k \in [r\theta]} \left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_{k-1}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{k}(\lambda) = 1\right]\right| \end{aligned} \tag{1}$$

Suppose, by contradiction, that there exists an adversary $\mathcal{A}$ that can distinguish between Game_{k-1} and Game_k with non-negligible probability. In Fig. 12 we construct an adversary $\mathcal{A}_0$ against Com'' that uses $\mathcal{A}$ as a subroutine to break the hiding of Com''.

Adversary $\mathcal{A}_0$

1. On input 1^λ, invoke $\mathcal{A}$ and receive messages $\mathsf{m}_0, \mathsf{m}_1 \in \{0,1\}^m$.
2. Sample a random bit $b \leftarrow\!\$\ \{0,1\}$ (this corresponds to the bit used in the games Game_{k-1} and Game_k).
   Computation of $\mathsf{a}_1, \ldots, \mathsf{a}_r$.
   (a) Let (j^*, i^*) be the indices corresponding to the k-th commitment pair.
   (b) Compute commitment tuples where the first $k-1$ pairs use shares of m_{1-b} and the pairs from $k+1$ up to $r\theta$ use shares of m_b.
   (c) For the k-th pair, sample $s^0_{j^*,i^*} \in \{0,1\}^m$ uniformly, compute $\tau^0_{j^*,i^*}$ as a commitment to $s^0_{j^*,i^*}$, and set $n_0 = \mathsf{m}_b \oplus s^0_{j^*,i^*}$ and $n_1 = \mathsf{m}_{1-b} \oplus s^0_{j^*,i^*}$.
   (d) Output (n_0, n_1) as the challenge messages to the challenger.
   (e) Upon receiving a commitment τ^* from the challenger (that depends on the unknown uniform bit b^*), use $(\tau^0_{j^*,i^*}, \tau^*)$ as the k-th commitment pair.
3. Store $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$, where the i^*-th commitment pair of a_{j^*} is equal to $(\tau^0_{j^*,i^*}, \tau^*)$.
   Computation of all c_j and z_j values.
4. For every $j \in [r]$:
   (a) If $j \neq j^*$: compute c_j and z_j according to $\mathsf{Com.Sen}'$ in Fig. 5.
   (b) If $j = j^*$:
      i. Set $\mathcal{E}_{j^*} = \varnothing$.
      ii. Sample $\mathsf{c}_{j^*} \leftarrow\!\$\ \{0,1\}^\theta \setminus \mathcal{E}_{j^*}$.
      iii. If $\mathsf{c}_{j^*,i^*} \neq 0$ abort the simulation for $\mathcal{A}$, and output a random bit to the challenger.
      iv. Compute z_{j^*} according to Step 2c of $\mathsf{Com.Sen}'$ in Fig. 5.

[21] We index the commitment pairs linearly by flattening all the r rounds, e.g., for $j = 3$ and $i = 1$, the commitment pair is the $(3\theta + 1)$-th pair.

v. Given msg be the tuple $(A, j^*, \mathsf{c}_{j^*}, \mathsf{z}_{j^*})$, interact with $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with input $(\mathsf{HashQuery}, (\mathsf{id}, \mathsf{msg}))$, upon receiving the tuple $(\mathsf{HashConfirm}, h)$ check if the first ℓ-bits of h are different from 0, if so set $\mathcal{E}_{j^*} \leftarrow \mathcal{E}_{j^*} \cup \{\mathsf{c}_{j*}\}$ and go to Step 4(b)ii.

5. Otherwise, complete the simulation, output $\{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ to $\mathcal{A}$, and output $\mathcal{A}$'s guess.

Fig. 12: Adversary $\mathcal{A}_0$ against the hiding property of Com'', constructed from an adversary $\mathcal{A}$ that distinguishes between Game_{k-1} and Game_k.

First, observe that $\mathcal{A}_0$ aborts the simulation only when $\mathsf{c}_{j^*,i^*} \neq 0$. When $\mathcal{A}_0$ does not abort and the challenger's bit $b^* = 0$, the commitment τ^* is to $n_0 = \mathsf{m}_b \oplus s^0_{j^*,i^*}$. In this case, both components of the i^*-th pair in a_{j^*} are commitments to shares that reconstruct to m_b. Since the first $k-1$ pairs use shares of m_{1-b} and pairs $k+1$ through $r\theta$ use shares of m_b, $\mathcal{A}_0$ perfectly simulates Game_{k-1} to $\mathcal{A}$.

Conversely, when $\mathcal{A}_0$ does not abort and $b^* = 1$, the commitment τ^* is to $n_1 = \mathsf{m}_{1-b} \oplus s^0_{j^*,i^*}$. Here, the i^*-th pair in a_{j^*} consists of commitments to shares reconstructing to m_{1-b}. Thus, the first k pairs use shares of m_{1-b} and the remaining use shares of m_b, perfectly simulating Game_k to $\mathcal{A}$.

Let $\mathsf{E}_{\mathsf{Abort}}$ be the event that $\mathcal{A}_0$ aborts the simulation according to Step 4(b)iii and outputs a random bit. We denote $\Pr[\overline{\mathsf{E}}_{\mathsf{Abort}}] = p$. As previously stated, we assume that the difference between the two consecutive games $\left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_{k-1}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{k}(\lambda) = 1\right]\right|$ is non-negligible, which we denote as ε. We begin by observing:

$$\begin{aligned}&\left|\Pr\left[\mathsf{Game}^{\mathcal{A}}_{k-1}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathcal{A}}_{k}(\lambda) = 1\right]\right| = \\ &\left|\Pr\left[\mathrm{HDN}^{\mathcal{A}_0}_{\mathsf{Com}''}(\lambda) = 1 \;\middle|\; b^* = 0, \overline{\mathsf{E}}_{\mathsf{Abort}}\right] - \Pr\left[\mathrm{HDN}^{\mathcal{A}_0}_{\mathsf{Com}''}(\lambda) = 1 \;\middle|\; b^* = 1, \overline{\mathsf{E}}_{\mathsf{Abort}}\right]\right| \geq \varepsilon\end{aligned} \tag{2}$$

This follows from the fact that $\mathcal{A}_0$ outputs the same result as $\mathcal{A}$ when there is no abort and, as established earlier, simulates the $(k-1)$-th game in one case and the k-th game in the other. Note that since the difference involves two probabilities (each between 0 and 1), we are sure that also their absolute sum is at least ε.

Now, considering that $\mathrm{HDN}^{\mathcal{A}_0}_{\mathsf{Com}''}(\lambda) = 1$ when $b^* = v$ for $v \in \{0,1\}$ means that $\mathcal{A}_0$ outputs v, by the law of total probability we have:

$$\begin{aligned}\forall v \in \{0,1\} : \Pr[\mathcal{A}_0 = v \mid b^* = v] = \\ \Pr\left[\mathcal{A}_0 = v \;\middle|\; b^* = v, \overline{\mathsf{E}}_{\mathsf{Abort}}\right] \Pr\left[\overline{\mathsf{E}}_{\mathsf{Abort}} \;\middle|\; b^* = v\right] \\ + \Pr[\mathcal{A}_0 = v \mid b^* = v, \mathsf{E}_{\mathsf{Abort}}] \Pr[\mathsf{E}_{\mathsf{Abort}} \mid b^* = v]\end{aligned}$$

Since the probability of event $\mathsf{E}_{\mathsf{Abort}}$ is independent of b^* and $\mathcal{A}_0$ guesses uniformly over $\{0,1\}$ when $\mathsf{E}_{\mathsf{Abort}}$ occurs, we obtain:

$$\forall v \in \{0,1\} : \Pr[\mathcal{A}_0 = v \mid b^* = v] = \Pr\left[\mathcal{A}_0 = v \;\middle|\; b^* = v, \overline{\mathsf{E}}_{\mathsf{Abort}}\right] p + 2^{-1}(1-p)$$

By solving for $\Pr\left[\mathcal{A}_0 = v \;\middle|\; b^* = v, \overline{\mathsf{E}}_{\mathsf{Abort}}\right]$ and substituting into Eq. (2), we directly derive:

$$\begin{aligned}&(1/p)|\Pr[\mathcal{A}_0 = 0 \mid b^* = 0] + \Pr[\mathcal{A}_0 = 1 \mid b^* = 1] - (1-p)| = \\ &(2/p)|(1/2)\Pr[\mathcal{A}_0 = 0 \mid b^* = 0] + (1/2)\Pr[\mathcal{A}_0 = 1 \mid b^* = 1] - 1/2 - p/2| \geq \varepsilon\end{aligned}$$

By removing $-|p/2|$ from both sides and applying the triangle inequality, and recalling that $\mathrm{HDN}^{\mathcal{A}_0}_{\mathsf{Com}''}(\lambda) = 1$ when $b^* = v$ for $v \in \{0,1\}$ means $\mathcal{A}_0$ outputs v, we obtain:

$$\mathsf{Adv}^{\mathrm{hdn}}_{\mathsf{Com}''}(\mathcal{A}_0, \lambda) = \left|\Pr\left[\mathrm{HDN}^{\mathcal{A}_0}_{\mathsf{Com}''}(\lambda) = 1\right] - 1/2\right| \geq \left(\frac{p\varepsilon}{2} - \frac{p}{2}\right) = \frac{p}{2}(\varepsilon - 1) \tag{3}$$

To conclude the proof, we need to show that $(p/2)(\varepsilon - 1)$ is non-negligible. This reduces to proving that p is non-negligible because we assumed that ε is non-negligible, and the only factor that could make this quantity negligible is p. Recall that p is the probability of the event $\overline{\mathsf{E}}_{\mathsf{Abort}}$. We now argue why the probability that $\mathcal{A}_0$ does not abort is non-negligible.

Consider a less powerful adversary $\mathcal{A}_0{}'$ that behaves exactly as $\mathcal{A}_0$ except for one difference: after Step 2d, during the sampling of the challenge c_{j^*}, instead of sampling (and trying) a new challenge c_{j^*} and

potentially succeeding in the simulation, $\mathcal{A}_0{}'$ aborts and outputs a random bit. Clearly, the probability of $\mathcal{A}_0$ not aborting is at least equal to the probability of $\mathcal{A}_0{}'$ not aborting, since $\mathcal{A}_0{}'$'s strategy is included in $\mathcal{A}_0$'s strategy, that is, $\mathcal{A}_0$ has multiple opportunities to avoid aborting while $\mathcal{A}_0{}'$ has only one.

The probability of $\mathcal{A}_0{}'$ not aborting is at least the probability that the first $\mathsf{c}_{j^*,i^*} \neq 0$ (which occurs with probability 1/2) and that the output of query to the $\mathcal{G}_{\mathsf{oRO}}$ functionality `Query` interface at Step 2d equals 0^ℓ (which occurs with probability $2^{-\ell}$). Since 2^ℓ is (lower than) the expected running time of the polynomial algorithm $\mathsf{Com.Sen}'$, the value $2^{-\ell}$ must be bounded by a polynomial $\mathsf{poly}(\lambda)$.

This completes the proof, as we have contradicted the assumption that the hiding property of Com'' holds by exhibiting $\mathcal{A}_0$ with a non-negligible advantage in the hiding game. By considering Eqs. (1) to (3), we obtain that the difference between two adjacent games is negligible (otherwise, we reach a contradiction), thus concluding the proof. □

E Proof of Thm. 2

We describe the strategy of the extractor $\mathsf{Com.Ext}$ in Fig. 13.

Extractor $\mathsf{Com.Ext}$

On input the string id a commitment τ such that can be successfully opened leveraging $\mathsf{Com.Rec}'$, and the query-answer list $\mathcal{Q}$ outputted by a call to the `Observe` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality, the algorithm $\mathsf{Com.Ext}$ executes the following steps:

1. Parse $\tau = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$, and set $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$.
2. For every query-answer tuple in the list $\mathcal{Q}$ of the form $(\mathsf{id}', \mathsf{query}, \mathsf{answer})$:
   (a) If $\mathsf{id}' \neq \mathsf{id}$ go to the next tuple in $\mathcal{Q}$.
   (b) If $\mathsf{query} = (A, j, \mathsf{c}^*, \mathsf{z}^*)$ and $(\mathsf{c}_j, \mathsf{z}_j) \neq (\mathsf{c}^*, \mathsf{z}^*)$:
      i. Parse $\mathsf{a}_j = (\tau^0_{j,1}, \tau^1_{j,1}), \ldots, (\tau^0_{j,\theta}, \tau^1_{j,\theta})$ and $\mathsf{c}^* = (\mathsf{c}^*_1 \| \ldots \| \mathsf{c}^*_\theta)$, $\mathsf{z}^* = (\mathsf{d}^{\mathsf{c}^*_1}_1, s^{\mathsf{c}^*_1}_1), \ldots, (\mathsf{d}^{\mathsf{c}^*_\theta}_\theta, s^{\mathsf{c}^*_\theta}_\theta)$ (the same is done for c_j and z_j according to the notation in Fig. 5).
      ii. If $(\mathsf{Com.Rec}''(\mathsf{d}^{\mathsf{c}^*_i}_i, \tau^{\mathsf{c}^*_i}_{j,i}, s^{\mathsf{c}^*_i}_i) = 1)$, for every $i \in [\theta]$; then search for the first $\mathsf{c}^*_i \neq \mathsf{c}_{j,i}$ and Output $\mathsf{m} = s^{\mathsf{c}^*_i}_i \oplus s^{\mathsf{c}_{j,i}}_{j,i}$.
3. Output $\perp$.

Fig. 13: The extractor $\mathsf{Com.Ext}$ for the commitment scheme Com'.

To prove that the extractor fails in extracting the message only with negligible probability when the commitment τ is well-formed and can be successfully decommitted, we define the event $\mathsf{E}_{\mathsf{ExtFail}}$. This event occurs when $\mathsf{Com.Ext}$ in Fig. 13 outputs $\perp$. More precisely, $\mathsf{E}_{\mathsf{ExtFail}}$ is the event in which $\mathcal{A}$ produces a commitment τ such that:

1. $\mathcal{A}$ can successfully decommit τ to some message m with non-negligible probability, and
2. $\mathcal{A}$ queries the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality with no more than a single accepting transcript for each $j \in [r]$[22].

Thus, it suffices to analyze the probability that $\mathsf{E}_{\mathsf{ExtFail}}$ happens and show that it is negligible in the security parameter. To do this, we first ignore all queries made to $\mathcal{H}$ that are not accepting transcripts, and then separate queries prefixed by different $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$ as they effectively instantiate independent ROs (i.e., it is like having a RO with the input also including a seed that differs according to A) and cannot be combined to produce a valid commitment τ.

Thus, for a fixed A, the event $\mathsf{E}_{\mathsf{ExtFail}}$ in which the adversary $\mathcal{A}$ can output an accepting commitment τ with fewer than 2 accepting transcripts (prefixed by A) queried to the `Query` interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality for each $j \in [r]$ is exactly the event that the first such accepting transcript queried to the $\mathcal{G}_{\mathsf{oRO}}$ functionality for every $j \in [r]$ outputs $(\mathsf{HashCompute}, h)$ such that the first ℓ-bit of h are 0.

[22]Note that we do not consider the case where the adversary queries more than one accepting transcript for some $j \in [r]$ but with the same challenge. This is because such queries are effectively identical (and count as one query). That is, the query input for the j-th computation is the tuple $(A, j, \mathsf{c}_j, \mathsf{z}_j)$ where A and j are fixed. If c_j is the same for a well-formed commitment, then z_j must also be the same. Thus, since the same input to the RO produces the same output, these are not distinct queries but repetitions of the same query.

This is equivalent to r independent uniformly chosen ℓ-bit strings being equal to 0, which happens with probability $(2^{-\ell})^r = 2^{-\lambda}$. For an adversary that makes $|\mathcal{Q}|$ queries to the $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality (where $|\mathcal{Q}| \in \mathbb{N}$ is the total number of queries made), the extraction abort probability is therefore bounded by $|\mathcal{Q}| \cdot 2^{-\lambda}$ that is a negligible value in the security parameter. □

Over-extraction of the commitment. We recall now what it means for an extractable commitment scheme to suffer from over-extraction. Following [GLOV12], over-extraction occurs when the extractor outputs a value different from $\perp$ for a commitment that is well-formed but has no valid opening. Our extractor $\mathsf{Com.Ext}$ in Fig. 13 clearly suffers from this. We highlight this because it requires particular care when using such commitments in our main result, both in the construction itself and in the corresponding security analysis.

Indeed, consider a commitment $\tau = \{(\mathsf{a}_j, \mathsf{c}_j, \mathsf{z}_j)\}_{j\in[r]}$ that is well-formed according to Fig. 5. That is, all queries to the $\mathtt{Query}$ interface of the $\mathcal{G}_{\mathsf{oRO}}$ functionality have ℓ heading zeros and for every $j \in [r]$, θ commitments in a_j correctly decommit in accordance with z_j. The extractor $\mathsf{Com.Ext}$ following the strategy in Fig. 13 outputs the XOR of the first pair of messages for which it has valid openings.

A (malicious) sender, however, could compute τ where the commitments in $A = (\mathsf{a}_1, \ldots, \mathsf{a}_r)$ do not commit to the shares of the same message $r\theta$ times. Instead, the sender could use a different message for each underlying pair of commitments, resulting in $r\theta$ pairs of commitments that commit to $r\theta$ pairs of strings whose XOR outputs different messages. This strategy becomes evident when looking at Step 1a in Fig. 5. Such a commitment τ is well-formed even though it can never be opened successfully, because the decommitment phase in Fig. 5 will certainly output 0.

F Instantiation via 1-to-1 OWFs

Our construction in Sec. 4.1 makes black-box use of a non-interactive commitment scheme. By the classical result of Blum [Blu81], a non-interactive commitment scheme that is perfectly binding and computationally hiding can be constructed under the assumption that 1-to-1 OWFs exist. Combining this with Thms. 1 and 2, we obtain the following corollary.

Corollary 4. *Assuming the existence of 1-to-1 OWFs, there exists a perfectly binding, non-interactive, straight-line extractable, and provable commitment scheme considering the $\mathcal{G}_{\mathsf{oRO}}$ functionality, according to Def. 5.*

The corollary follows immediately by instantiating the commitment scheme Com'' in Fig. 5 with Blum's perfectly binding commitment scheme [Blu81].

One might observe that Blum's scheme yields a non-interactive bit commitment that is straight-line extractable and provable. As such, to commit to an entire message, one must commit in parallel to each bit individually. Since the size of the commitment already depends on the security parameter, this could result in a prohibitive size. This issue can be mitigated using a standard technique (see [KLP22, Section A.3]), originally proposed by Naor [Nao91], leveraging a PRG. In a nutshell, to commit to a message of arbitrary length, one runs the bit commitment scheme in parallel over a seed of a PRG, and then uses the output of the PRG as a one-time pad for the message.

G Comparison of Blind Signature Functionalities

While the concept of blind signatures has been well-established in the literature since Chaum introduced it 40 years ago, there is no widely-used/standard functionality that idealizes the behavior of blind signature protocols. First, most works focus on the (less demanding) game-based definition of blind signatures[23]; and second, several different formalizations of blind signature functionalities exist in the literature.

To the best of our knowledge, there are at least three distinct functionalities that have been proposed. The first was introduced by Lindell [Lin03, Lin08] to prove the impossibility of (self-)concurrent composition of blind signatures in the plain model. Since Lindell's objective was not to achieve a UC-secure construction but rather to establish an impossibility result, his functionality naturally lacks UC-related

[23] This focus on game-based definitions can be attributed to several impossibility results that limit the applicability of blind signatures in real-world scenarios. For example, Lindell demonstrated in [Lin03, Lin08] that is impossible to achieve concurrent blind signatures in the plain model with simulation-based security and black-box simulation, while it is known that such protocols are possible under game-based definitions [HKKL07].

details (e.g., there is no mention of session or party identifiers). In a nutshell, Lindell's functionality [Lin08, Section 6.1] stays close to Chaum's original definition, which focused on blind signature issuance for already deployed signature schemes (i.e., RSA in Chaum's case). This functionality is thus based on the idea of a 2PC that runs the signature generation algorithm of an underlying signature scheme.

The second functionality is from Fischlin's work [Fis06a], which was used to achieve UC-secure and round-optimal blind signatures in the common reference string model. This functionality is based on Canetti's signature functionality [Can03] and, according to Fischlin, it faithfully preserves its structure while introducing only minimal modifications necessary to capture the blind signature setting. The functionality more closely aligns with the modern understanding of blind signatures, as it is not bound to any underlying signature scheme and avoids embedding cryptographic algorithms within the functionality itself.

The third functionality comes from Kiayias and Zhou [KZ08, KZ07], which broadly follows Fischlin's functionality but is instead based on a different signature functionality also due to Canetti (we provide more details about this functionality later). Furthermore, in the full version of their work, the authors claim to achieve UC security for their blind signature construction with respect to adaptive corruption, whereas Fischlin assumes only non-adaptive corruption.

We note that other works [AO09, CLZ04] also consider UC blind signatures, but they formalize functionality that differs from the classical blind signature notion. These works capture alternative security properties (i.e., easy to compute according to the authors) rather than the standard security properties reflected in traditional game-based definitions.

Rationale for adopting Fischlin's functionality Unfortunately, none of the three works that propose different blind signature functionalities include a comparison, so it is unclear which one is best suited for use. We now explain why we chose to adopt Fischlin's functionality over the others.

Lindell's Functionality. Lindell's functionality was proposed at a time when blind signatures were mostly seen as running a signature scheme inside a 2PC. As a result, the functionality is designed for that setting and includes cryptographic algorithms, meaning it depends on the existence of a secure signature scheme. However, this kind of approach is usually avoided when defining ideal functionalities. Moreover, nowadays blind signature schemes (e.g., see [TZ22, HLW23, CATZ24, KRW24]) are not necessarily based on pre-existing signature schemes. To use Lindell's functionality with such protocols would require first defining an underlying signature scheme (which may not naturally exist), creating an unnecessary complication. While one could trivially derive a signature scheme by having the signer simulate the 2PC of the blind signature protocol, this approach is cumbersome. Restricting to blind signatures that output a signature for a standard signature scheme imposes an unnecessarily strict requirement that is neither needed nor achieved in our work.

Additionally, Lindell's functionality aims to provide security even when the signer (the party possessing the secret key) is honest but uses potentially malformed/malicious keys provided by the environment. While this might be an appealing feature, de facto it requires security even when the honest signer uses badly generated keys. Note that even standard signature schemes do not guarantee unforgeability in such cases.

Kiayias and Zhou's Functionality. Kiayias and Zhou primarily focus on equivocal blind signatures to handle adaptive corruptions, which fall outside the scope of our work. Moreover, they base the functionality on a version of Canetti's signature functionality that differs from the one in [Can03] that has been "withdrawn" by Canetti himself later[24], while the version on which Fischlin based the UC functionality is still present and widely used today. In general, the functionalities of Fischlin and the one of Kiayias and Zhou are very close, differing only in which signature functionality they take as a starting point. With respect to adaptive corruption, this is indeed an interesting direction to explore. However, in this work, we first focus on achieving GUC-secure blind signatures, considering only NPRO and minimal assumptions, leaving adaptive security and equivocal blind signatures as a potential direction for future work.

[24]Note that the version that the authors refer to is the functionality presented in the 2005 version of Canetti's UC paper [Can00, TR01-016] but later removed in the revised version of 2013 and is no longer present in the published version [Can20].

H Description of Fischlin's Blind Signature Functionality

As previously mentioned, $\mathcal{F}_{\mathsf{BS}}$ closely follows the regular signature functionality of Canetti [Can03], adapting it to the blind signature setting.

In a nutshell, the functionality allows the adversary to select the public key. For verification requests involving signatures generated by the functionality, acceptance is guaranteed; in all other cases, the adversary determines whether a given signature σ for a message m is considered valid.

Regarding signature requests, the adversary must not learn the messages of honest users, and signatures must not be linkable to their signing requests, reflecting the blindness property. To ensure this, the adversary describes a stateless, possibly randomized, algorithm BlSig to the blind signature functionality. The algorithm is declared and fixed during the key generation phase, when the adversary chooses the public key and it is then used in all subsequent signature generation instances. Whenever an honest user requests a signature for a message m, the functionality invokes $\mathsf{BlSig}(\mathsf{m})$ to produce a signature σ, without revealing m to the adversary, thereby maintaining blindness. We additionally allow the signer to explicitly abort the signing process, a behavior that is only implicit in Fischlin's original formulation.

In case of corrupt users, however, the functionality does not run BlSig. Instead, it asks the adversary to output the possible signature σ that the corrupt user could obtain through interaction with the signer. Importantly, a corrupt user may not necessarily output a signature immediately (or ever) following such an interaction; the interaction merely ensures that the user could, in principle, generate the signature. Consequently, the functionality does not return the adversary's potential signature σ to the corrupt user. Finally, for any signature request, the functionality informs the signer of the request, but does not disclose the actual message m or the resulting signature σ.

Note that the functionality captures the standard security guarantee of blind signatures known as one-more unforgeability, which ensures that it is impossible to create a signature for a message that was never queried to the blind signature functionality. We observe that a stronger security notion exists where, beyond preventing forgeries for unqueried messages, it also requires that an adversary cannot produce more than one valid signature for any queried message[25]. This property, known as strong one-more unforgeability, is not guaranteed by Fischlin's functionality. In Fischlin's full version [Fis06b], he presents an alternative functionality and a construction that achieves this security guarantee, at the cost of stronger hardness assumptions. In this paper, we focus only on $\mathcal{F}_{\mathsf{BS}}$.

I Proof of Thm. 3

We proceed directly to the description of the simulator and the proof of security.

The simulator Sim.

- **Simulating the communication with the environment** $\mathcal{Z}$: The simulator Sim forwards all input values received from $\mathcal{Z}$ to $\mathcal{A}$'s input tape (as if originating from $\mathcal{Z}$) and likewise forwards all outputs from $\mathcal{A}$ to $\mathcal{Z}$.
- **Simulating the key generation step:** Upon receiving $(\mathsf{KGQuery}, \mathsf{sid})$ from $\mathcal{F}_{\mathsf{BS}}$, where $\mathsf{sid} = (\mathcal{S}, \mathsf{sid}')$ for some signer $\mathcal{S}$, the simulator Sim proceeds as follows.
  1. If $\mathcal{S}$ is honest, the simulator Sim samples randomness r and computes a key pair $(\mathsf{sk}', \mathsf{pk}')$ for the signature scheme Sig by running $\mathsf{Sig.KG}$ on input the security parameter in unary form with randomness r. Sim runs the algorithm $\mathsf{Com.Sen}$ on input the message 0, and obtain the commitment $\tau_{\mathcal{S}}$ and the decommitment information $\mathsf{d}_{\mathcal{S}}$. Sim sets $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ and $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$. Sim defines BlSig as a trivial algorithm that returns $\perp$ on any input message m (note that BlSig is never invoked when $\mathcal{S}$ is honest, according to $\mathcal{F}_{\mathsf{BS}}$).
  2. If $\mathcal{S}$ is corrupted, Sim sends $(\mathsf{KGQuery}, \mathsf{sid})$ to $\mathcal{A}$ and waits to receive $(\mathsf{VerificationKey}, \mathsf{sid}, \mathsf{pk})$. Sim checks if $\mathsf{sid} = (\mathcal{S}, \mathsf{sid}')$; if not, it does nothing and terminates. Otherwise, given $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$, where $\tau_{\mathcal{S}}$ is a commitment according to Com, Sim does the following:
     (a) Sends $(\mathsf{Observe}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{oRO}}$ and receives $(\mathsf{ListObserve}, Q_{\mathsf{sid}})$, where Q_{sid} is the list of all queries and answers made to the RO.

[25] Specifically, this stronger notion also prevents an adversary holding a valid message-signature pair (m, σ) from generating a different valid signature σ' for the same message m.

(b) Runs the extraction algorithm $\mathsf{Com.Ext}$ of Com as described in Fig. 13 on input commitment $\tau_{\mathcal{S}}$ and the list of RO queries Q_{sid}. If extraction fails (i.e., the extractor outputs $\perp$), Sim defines BlSig as a trivial algorithm that returns $\perp$ on any input message m.
Otherwise, Sim obtains the message $(\mathsf{sk}'\|r)$ that must be the unique message to which $\tau_{\mathcal{S}}$ can be correctly decommitted.

(c) If the extraction from $\tau_{\mathcal{S}}$ succeeds, then Sim sets sk equal to the tuple $(\mathsf{sk}', r, \mathsf{d}^*)$, where d^* is a length-matching string of decommitment information for the Com scheme, and Sim defines the behavior of algorithm BlSig on input message m as follows:

i. Run $\mathsf{Niwi.Prove}$ for relation $\mathsf{R}_{\mathcal{U}}$ (defined in Fig. 7) on statement $(\mathsf{pk}, \mathsf{m})$ using the (trapdoor) witness consisting of the secret key sk (appropriately padded with dummy values to match the required witness format) and obtain proof π.

ii. Output the signature σ equals to π.

Finally, in both cases, the simulator Sim sends $(\mathsf{VerificationKey}, \mathsf{sid}, \mathsf{pk}, \mathsf{BlSig})$ to $\mathcal{F}_{\mathsf{BS}}$ and distributes pk to all parties, as they expect to receive it from the signer $\mathcal{S}$.

– **Simulating the signature generation when the signer $\mathcal{S}$ is corrupted and the user $\mathcal{U}$ is honest**: Upon receiving $(\mathsf{SignDone}, \mathsf{sid})$ from $\mathcal{F}_{\mathsf{BS}}$, the simulator Sim:

1. Computes a commitment $\tau_{\mathcal{U}}$ to message 0 using algorithm $\mathsf{Com.Sen}$, then provides $\tau_{\mathcal{U}}$ to $\mathcal{A}$ as if sent from $\mathcal{U}$.
2. Receives (σ', π') from $\mathcal{A}$ and verifies that (a) the proof π' is valid by running $\mathsf{Niwi.Verify}$ for relation $\mathsf{R}_{\mathcal{S}}$ on statement $(\tau_{\mathcal{U}}, \mathsf{pk})$, and (b) the signature σ' is valid by running $\mathsf{Sig.Verify}$ on message $\tau_{\mathcal{U}}$ with public key pk.
3. If either check fails, Sim sends $(\mathsf{AgreeSignDone}, 0)$ to $\mathcal{F}_{\mathsf{BS}}$. Otherwise, it sends $(\mathsf{AgreeSignDone}, 1)$ to $\mathcal{F}_{\mathsf{BS}}$ and then $\mathcal{U}$ receives from $\mathcal{F}_{\mathsf{BS}}$ the tuple $(\mathsf{Signature}, \mathsf{sid}, \mathsf{m}, \sigma)$, where σ is generated by $\mathcal{F}_{\mathsf{BS}}$ using BlSig.

– **Simulating the signature generation when the signer $\mathcal{S}$ is honest and the user $\mathcal{U}$ is corrupted**: Upon receiving $\tau_{\mathcal{U}}$ from $\mathcal{A}$ (intended as a message from $\mathcal{U}$ to $\mathcal{S}$) in the session sid, the simulator Sim:

1. Verifies that $\tau_{\mathcal{U}}$ is a well-formed commitment as described in Fig. 5; ignores if this check fails.
2. Sends $(\mathsf{Observe}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{oRO}}$ and receives $(\mathsf{ListObserve}, Q_{\mathsf{sid}})$, where Q_{sid} is the list of all queries and answers made to the RO.
3. Runs the extraction algorithm $\mathsf{Com.Ext}$ of Com as described in Fig. 13 on input commitment $\tau_{\mathcal{U}}$ and the list of RO queries Q_{sid}. If extraction fails (i.e., the extractor outputs $\perp$), abort the simulation. Otherwise, Sim obtains a message m that must be the unique message to which $\tau_{\mathcal{U}}$ can be successfully decommitted. Sim also stores the (commitment) pairs extracted in Step 2(b)ii of Fig. 13 while running the $\mathsf{Com.Ext}$ algorithm: $(\mathsf{d}_A, \tau_A, s_A)$ and $(\mathsf{d}_B, \tau_B, s_B)$, where $\mathsf{Com.Rec}'(\mathsf{d}_A, \tau_A, s_A) = \mathsf{Com.Rec}'(\mathsf{d}_B, \tau_B, s_B) = 1$ and $s_A \oplus s_B = \mathsf{m}$.
4. Sends $(\mathsf{SignQuery}, \mathsf{sid}, \mathsf{m}, \mathsf{pk})$ to $\mathcal{F}_{\mathsf{BS}}$ and waits to receive $(\mathsf{SignQuery}, \mathsf{sid}, \mathsf{m})$ from $\mathcal{F}_{\mathsf{BS}}$. Upon receiving this message, Sim samples a uniform string σ that matches the length of the signature computed by an honest $\mathcal{U}$ in Step 2c of the interactive protocol in Fig. 8, and then sends $(\mathsf{Signature}, \mathsf{sid}, \mathsf{m}, \sigma)$ to $\mathcal{F}_{\mathsf{BS}}$.
5. Uses the signing key sk' from sk to compute signature σ' on message $\tau_{\mathcal{U}}$ received from $\mathcal{A}$ by running algorithm $\mathsf{Sig.Sign}$. Finally, Sim runs $\mathsf{Niwi.Prove}$ for relation $\mathsf{R}_{\mathcal{S}}$ (see Fig. 7) on statement $(\tau_{\mathcal{U}}, \mathsf{pk})$ with (trapdoor) witness consisting of the extracted values $s_A, s_B, \mathsf{d}_A, \mathsf{d}_B$ and length-matching values to match the witness structure, obtaining π'. Sends the tuple (σ', π') to $\mathcal{A}$, as expected from $\mathcal{S}$.

– **Simulating the signature verification step:** Upon receiving $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$ from $\mathcal{F}_{\mathsf{BS}}$, the simulator Sim: (1) parses σ as π, (2) computes b as the output of $\mathsf{Niwi.Verify}(\mathsf{stm}, \pi)$ for relation $\mathsf{R}_{\mathcal{S}}$, where the statement $\mathsf{stm} = (\mathsf{pk}, \mathsf{m})$, and (3) sends $(\mathsf{VerifyDone}, \mathsf{sid}, \mathsf{m}, \sigma, b)$ to $\mathcal{F}_{\mathsf{BS}}$.

Simulation in the case where both $\mathcal{S}$ and $\mathcal{U}$ are honest is straightforward. The simulator's knowledge of sk directly enables this simulation without additional complexity.

Analysis of the simulation. Recalling that ϕ_{BS} runs in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model, we need to prove that for every $\mathcal{A}$ and every $\mathcal{Z}$, $\mathrm{GEXEC}_{\phi_{\mathsf{BS}},\mathcal{A},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathrm{GIDEAL}_{\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$ according to Def. 12. We prove this via a series of hybrid games. For each hybrid game Game_i where $i \in \mathbb{N}$, we write $\mathsf{Game}_{i\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$ to denote the output of environment $\mathcal{Z}$ when interacting with the blind signature functionality $\mathcal{F}_{\mathsf{BS}}$ and simulator Sim in the i-th hybrid world (as in App. B).

$\underline{\mathsf{Game}_0}$. In this game, we modify GIDEAL by adding an extra extraction step after receiving the proof π' from the adversary $\mathcal{A}$. Specifically, when simulating signature generation for a corrupted signer $\mathcal{S}$ and honest user $\mathcal{U}$, we alter the behavior as follows. While in GIDEAL the simulator Sim receives the pair (σ', π'), checks the correctness of both σ' and π', and then sends $(\mathsf{AgreeSignDone}, 1)$ to $\mathcal{F}_{\mathsf{BS}}$, now before sending such message, the simulator Sim performs the following additional steps:

1. Sends $(\mathsf{Observe}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{oRO}}$ and receives $(\mathsf{ListObserve}, Q_{\mathsf{sid}})$, where Q_{sid} is the list of all queries and answers made to the random oracle.
2. Runs $\mathsf{Niwi.Ext}$ on input statement $\mathsf{stm} = (\tau_{\mathcal{U}}, \mathsf{pk})$, the proof π', and the list Q_{sid}, obtaining as output a witness wts.

The simulator aborts (i.e., outputs $\perp$) if $\mathsf{R}_{\mathcal{S}}(\mathsf{stm}, \mathsf{wts}) \neq 1$; otherwise, it continues the simulation exactly as in GIDEAL.

We now prove that GIDEAL is computationally indistinguishable from Game_0, based on the argument of knowledge property of the NIWI argument of knowledge system $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ described in Sec. 3.1. Note that the only case where the two games differ is when Sim outputs $\perp$; otherwise, the two games are identical and thus indistinguishable. Therefore, we need to analyze the probability with which such an abort occurs.

Assume by contradiction that Sim outputs $\perp$ with non-negligible probability. The reduction to the argument of knowledge property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ is immediate. Specifically, suppose there exists an environment $\mathcal{Z}$ that distinguishes between GIDEAL and Game_0 with non-negligible probability (because Sim outputs $\perp$ with non-negligible probability). We construct an adversary $\mathcal{A}_0$ against the argument of knowledge property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ as follows: $\mathcal{A}_0$ internally emulates the entire execution, but when the simulator would output $\perp$, $\mathcal{A}_0$ instead outputs the tuple (stm, π') to its challenger in the argument of knowledge game according to Fig. 11, where stm equals to $(\tau_{\mathcal{U}}, \mathsf{pk})$. It is immediate to see that $\mathcal{A}_0$ wins with exactly the same probability that $\mathcal{Z}$ distinguishes between the two games, which equals the probability that Sim outputs $\perp$. Since $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ satisfies the argument of knowledge property by assumption, this probability must be negligible. Finally, we note that according to the discussion in Sec. 2, we presented a definition of argument of knowledge (see Def. 11) that enables security against malleability attacks. Specifically, we prevent an adversary $\mathcal{A}$ from exploiting WI proofs computed in other sessions (i.e., leveraging RO queries made in other sessions) to counter successful extraction. This security is achieved by carefully incorporating the session identifier sid in queries to the $\mathcal{G}_{\mathsf{oRO}}$ functionality. Thus, we conclude that $\mathrm{GIDEAL}_{\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathsf{Game}_{0\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

$\underline{\mathsf{Game}_1}$. In this game, we modify Game_0 only in how the commitment $\tau_{\mathcal{U}}$ is computed when simulating signature generation for a corrupted signer $\mathcal{S}$ and honest user $\mathcal{U}$. While in Game_0 the simulator Sim computes a commitment to the message 0 (according to Step 1), in Game_1 the simulator instead computes a commitment to the actual message m, exactly as an honest user $\mathcal{U}$ would. That is, $\tau_{\mathcal{U}}$ becomes a commitment to m generated using algorithm $\mathsf{Com.Sen}$, with $\mathsf{d}_{\mathcal{U}}$ being the associated decommitment information that Sim stores.

We now prove that Game_0 is computationally indistinguishable from Game_1, based on the hiding property of the commitment scheme Com described in Def. 3. Assume by contradiction that there exists an environment $\mathcal{Z}$ that distinguishes between Game_0 and Game_1 with non-negligible probability. We construct an adversary $\mathcal{A}_1$ against the hiding property of Com as follows. $\mathcal{A}_1$ receives the public parameters for the commitment scheme from the hiding experiment challenger. $\mathcal{A}_1$ then internally emulates the entire execution. The only difference is that when Sim needs to create the commitment $\tau_{\mathcal{U}}$, $\mathcal{A}_1$ submits the pair $(0, \mathsf{m})$ to the hiding experiment challenger. Upon receiving a commitment from the challenger, $\mathcal{A}_1$ uses it as $\tau_{\mathcal{U}}$ in the simulation.

$\mathcal{A}_1$ continues the simulation until $\mathsf{Niwi.Ext}$ is executed and outputs a witness wts as in Game_0. Note that when $\mathsf{R}_{\mathcal{S}}(\mathsf{stm}, \mathsf{wts}) \neq 1$ where $\mathsf{stm} = (\tau_{\mathcal{U}}, \mathsf{pk})$, we can rely on our analysis in Game_0. By the argument presented there, this event occurs only with negligible probability. If it were to occur with non-negligible probability, we could directly apply the same arguments used in Game_0 to derive a contradiction. $\mathcal{A}_1$ checks the witness $\mathsf{wts} = (\mathsf{m}_0, \mathsf{m}_1, \mathsf{d}_0, \mathsf{d}_1, \mathsf{sk})$ and specifically it checks if a trapdoor witness has been extracted, namely, if there exist indices $j \in [r]$ and $i \in [\theta]$ such that commitments $\tau^0_{j,i}$ and $\tau^1_{j,i}$ (part of a pair from a_j which is part of $\tau_{\mathcal{U}}$ previously output by the challenger) can be successfully opened to messages m_0 and m_1 using d_0 and d_1, respectively. If this is the case, $\mathcal{A}_1$ halts the execution and computes $\mathsf{m}^* = \mathsf{m}_0 \oplus \mathsf{m}_1$, outputting 0 if $\mathsf{m}^* = 0$ and 1 if $\mathsf{m}^* = \mathsf{m}$[26].

[26]Note that by construction of $\tau_{\mathcal{U}}$ (i.e., correctly computed by the hiding oracle challenger), it is not possible that $\tau_{\mathcal{U}}$ suffers from over-extraction, nor for m^* to be different from both 0 and m.

If no tuple in $\tau_{\mathcal{U}}$ exists that can be successfully decommitted as described, $\mathcal{A}_1$ continues the simulation until $\mathcal{Z}$ produces its output, at which point $\mathcal{A}_1$ outputs the same guess.

Observe that when the challenger commits to 0, $\mathcal{A}_1$ perfectly simulates Game_0; when it commits to m, $\mathcal{A}_1$ perfectly simulates Game_1. When the extracted witness from π' is the trapdoor witness, $\mathcal{A}_1$ directly breaks the hiding property of Com. Otherwise, $\mathcal{A}_1$ distinguishes between the two games with exactly the same probability as $\mathcal{Z}$. Since Com satisfies the hiding property (as proved in Thm. 1), this probability must be negligible. Finally, we note that according to the discussion in Sec. 2, we presented a definition of hiding (see Def. 3) that enables security also against malleability attacks. Specifically, we consider an adversary $\mathcal{A}$ that can request the random oracle to see all the queries made to the RO of the form $(\mathsf{sid}', \mathsf{sid}||x)$ such that $\mathsf{sid}' \neq \mathsf{sid}$ (i.e., the illegitimate queries). Even with this capability, similar to what an extractor would have, the adversary still cannot succeed in breaking the hiding property of the commitment. Thus, we conclude that $\mathsf{Game}_{0\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathsf{Game}_{1\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

$\underline{\mathsf{Game}_2}$. In this game, we analyze the extraction of the message $(\mathsf{sk}'\|r)$ performed by Sim from the commitment $\tau_{\mathcal{S}}$ (which forms part of $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$) during the key generation phase, namely, before the definition of the BISig algorithm from Sim. We show that, provided the commitment can be successfully opened (i.e., is well-formed), the extraction procedure fails only with negligible probability.

First, we note that according to the discussion in Sec. 2, we presented a definition of straight-line extractability (see Def. 4) that enables security against malleability attacks. Specifically, we prevent an adversary $\mathcal{A}$ from exploiting commitment computed in other sessions (i.e., leveraging RO queries made in other sessions) to counter successful extraction. This security is achieved by carefully incorporating the session identifier sid in queries to the $\mathcal{G}_{\mathsf{oRO}}$ functionality.

Next, observe that if the extraction from the commitment $\tau_{\mathcal{S}}$ of the committed message $(\mathsf{sk}'\|r)$ fails, then Sim must define BISig as a trivial algorithm that returns $\perp$ on any input message m. If BISig is subsequently invoked by the functionality, because $\mathcal{A}$ sends to Sim a tuple (σ', π') such that both the signature and π' can be correctly verified, then the "signature" will be $\perp$ for every message. This clearly allows the environment $\mathcal{Z}$ to distinguish the ideal-world from the real-world execution. Hence, we must show that this event occurs only with negligible probability. We refer to this event as $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$.

Assume, for contradiction, that $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ occurs with non-negligible probability. We present a reduction to the extractability property of the commitment scheme Com. Specifically, consider an adversary $\mathcal{A}_2$ who aims to leverage the event $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ to break the extractability property of the commitment scheme associated with $\tau_{\mathcal{S}}$.

Adversary $\mathcal{A}_2$ proceeds by running Game_2 up to the point where $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ occurs. At this stage, $\mathcal{A}_2$ has extracted, as in Game_0, the witness $\mathsf{wts} = (\mathsf{m}_0, \mathsf{m}_1, \mathsf{d}_0, \mathsf{d}_1, \mathsf{sk})$ from π', and $\mathcal{A}_2$ is sure that π' was not computed using the "trapdoor" (part of the) witness $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{d}_0, \mathsf{d}_1)$ (i.e., $\mathcal{A}_2$ is sure that, to compute π', $\mathcal{A}$ used sk that is related to pk according to $\mathsf{R}_{\mathcal{S}}$), due to the arguments established in Game_1. At this point, $\mathcal{A}_2$ decodes sk as the tuple $(\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$, and since $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$, $\mathcal{A}_2$ returns to its challenger the tuple $(\mathsf{d}_{\mathcal{S}}, \tau_{\mathcal{S}}, \mathsf{sk}'\|r)$. Because the event $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ occurred, $\mathcal{A}_2$ is guaranteed to win the extractability game (i.e., $\mathcal{A}_2$ knows how to decommit $\tau_{\mathcal{S}}$, but the extractor fails to recover the message).

Thus, $\mathcal{A}_2$ wins the extractability game for the commitment scheme Com whenever $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ occurs (i.e., whenever $\tau_{\mathcal{S}}$ is well-formed but extraction fails). Since Com is assumed to satisfy extractability, this probability must be negligible. We therefore conclude that the probability of $\mathsf{E}^{\tau_{\mathcal{S}}}_{\mathsf{Ext}}$ is negligible; that is, the probability that extraction from a well-formed commitment $\tau_{\mathcal{S}}$ of the message $(\mathsf{sk}'\|r)$ outputs $\perp$ is negligible.

$\underline{\mathsf{Game}_3}$. In this game, following Game_2, we modify the behavior of the BISig algorithm. According to the description of BISig in the strategy of Sim (when simulating against a corrupted signer and an honest user), BISig runs the algorithm $\mathsf{Niwi.Prove}$ for the relation $\mathsf{R}_{\mathcal{U}}$ on the statement $(\mathsf{pk}, \mathsf{m})$, using the extracted secret key sk as the trapdoor witness to generate the proof π. Note that, as established in Game_2, whenever BISig is invoked (by the functionality), the secret key sk has been successfully extracted (except with negligible probability), and this remains an invariant in the current game.

In Game_3, we change the witness used by BISig. Instead of using the trapdoor witness sk, we now use the witness that would be used by an honest user $\mathcal{U}$, following the description in Fig. 8. Specifically, the BISig algorithm uses the decommitment information $\mathsf{d}_{\mathcal{U}}$, the signature σ' received from $\mathcal{A}$, and the commitment $\tau_{\mathcal{U}}$. That is, BISig internally runs the algorithm $\mathsf{Niwi.Prove}$ for the relation $\mathsf{R}_{\mathcal{U}}$ on input statement $(\mathsf{pk}, \mathsf{m})$ and witness $(\mathsf{d}_{\mathcal{U}}, \sigma', 1^{|\mathsf{sk}|}, \tau_{\mathcal{U}})$, exactly as an honest user $\mathcal{U}$ would.

We now prove that Game_2 is computationally indistinguishable from Game_3, based on the witness indistinguishability property of the NIWI argument of knowledge scheme $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ described in Def. 10.

Assume by contradiction that there exists an environment $\mathcal{Z}$ that distinguishes between Game_2 and Game_3 with non-negligible probability. We construct an adversary $\mathcal{A}_3$ against the WI property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ as follows.

$\mathcal{A}_3$ receives the public parameters for the NIWI scheme from the WI experiment challenger and internally emulates the entire execution of Game_2. When BlSig needs to generate the proof π, $\mathcal{A}_3$ prepares: (a) the trapdoor witness $\mathsf{wts}_0 = \mathsf{sk}$, and (b) the honest witness $\mathsf{wts}_1 = (\mathsf{d}_{\mathcal{U}}, \sigma', 1^{|\mathsf{sk}|}, \tau_{\mathcal{U}})$. Note that, except with negligible probability, $\mathcal{A}_3$ always obtains the trapdoor witness sk whenever BlSig is invoked. This follows from the argument established above: in Game_2, we demonstrated that the extraction of sk from the commitment $\tau_{\mathcal{S}}$ succeeds except with negligible probability, provided the commitment is well-formed. Thus, in such Game_3, whenever BlSig is called, the extracted secret key sk is guaranteed to be correct except with negligible probability, because what we proved in Game_2 remains invariant.

$\mathcal{A}_3$ then submits the statement $\mathsf{stm} = (\mathsf{pk}, \mathsf{m})$ along with both witnesses wts_0 and wts_1 to the WI challenger's proving oracle Prove. Upon receiving proof π from the challenger, $\mathcal{A}_3$ uses it as the final signature σ output by BlSig. $\mathcal{A}_3$ continues the simulation until $\mathcal{Z}$ produces its output, at which point $\mathcal{A}_3$ outputs the same guess.

Note that both witnesses satisfy the relation $\mathsf{R}_{\mathcal{U}}$. The witness wts_0 satisfies the relation through the trapdoor branch since pk is the public key corresponding to sk, and this is always true except with negligible probability, according to the above arguments. The witness $\mathsf{wts}_1 = (\mathsf{d}_{\mathcal{U}}, \sigma', 1^{|\mathsf{sk}|}, \tau_{\mathcal{U}})$ satisfies the relation because: (a) $\mathsf{d}_{\mathcal{U}}$ is a valid decommitment for $\tau_{\mathcal{U}}$ to message m, as constructed by Sim, and (b) σ' is a valid signature on $\tau_{\mathcal{U}}$ under the public key pk due to the perfect completeness of the Sig scheme.

When the WI challenger's bit equals 0 (using witness wts_0), $\mathcal{A}_3$ perfectly simulates Game_2; when it equals 1 (using witness wts_1), $\mathcal{A}_3$ perfectly simulates Game_3. Therefore, $\mathcal{A}_3$ breaks the WI property of Niwi with exactly the same probability that $\mathcal{Z}$ distinguishes between the two games.

Finally, we note that according to the discussion in Sec. 2, we presented a definition of WI (see Def. 10) that enables security also against malleability attacks. Specifically, we consider an adversary $\mathcal{A}$ that can request the random oracle to see all the queries made to the RO of the form $(\mathsf{sid}', \mathsf{sid}||x)$ such that $\mathsf{sid}' \neq \mathsf{sid}$ (i.e., the illegitimate queries). Even with this capability, similar to what an extractor would have, the adversary still cannot succeed in breaking the WI property of π.

Since Niwi satisfies the WI property, this probability must be negligible. Thus, we conclude that $\mathsf{Game}_{2\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathsf{Game}_{3\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

$\underline{\mathsf{Game}_4}$. In this game, we add an extra extraction step after receiving the final signature σ (i.e., the proof π) from the adversary $\mathcal{A}$ during the verification step. Specifically, when simulating signature generation for an honest signer $\mathcal{S}$ and a corrupted user $\mathcal{U}$, we alter the behavior as follows. That is, before sending $(\mathsf{VerifyDone}, \mathsf{sid}, \mathsf{m}, \sigma, b)$ to $\mathcal{F}_{\mathsf{BS}}$, the simulator Sim does the following:

1. Sends $(\mathsf{Observe}, \mathsf{sid})$ to $\mathcal{G}_{\mathsf{oRO}}$ and receives $(\mathsf{ListObserve}, Q_{\mathsf{sid}})$, where Q_{sid} is the list of all queries and answers made to the RO.
2. Runs $\mathsf{Niwi.Ext}$ on input statement stm (the tuple $(\mathsf{pk}, \mathsf{m})$), the proof π, and the list Q_{sid} to obtain a witness value wts.

The simulator aborts (i.e., outputs $\perp$) if $\mathsf{R}_{\mathcal{U}}(\mathsf{stm}, \mathsf{wts}) \neq 1$, otherwise continues the simulation exactly as in Game_3.

We now prove that Game_3 is computationally indistinguishable from Game_4, based on the argument of knowledge property of the NIWI argument of knowledge system $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ described in Sec. 3.1. Note that the only case where the two games differ is when Sim outputs $\perp$; otherwise, the two games are identical and thus indistinguishable. Therefore, it remains to analyze the probability with which such an abort happens. Assume by contradiction that Sim outputs $\perp$ with non-negligible probability. The reduction to the argument of knowledge property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ is immediate. Specifically, assume that there exists an environment $\mathcal{Z}$ that distinguishes between Game_3 and Game_4 with non-negligible probability (because Sim outputs $\perp$ with non-negligible probability). We construct an adversary $\mathcal{A}_4$ against the argument of knowledge property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ as follows: $\mathcal{A}_4$ internally emulates the entire execution, but when the simulator would output $\perp$, $\mathcal{A}_4$ instead outputs the tuple (stm, π) to its challenger in the argument of knowledge game according to Fig. 11. $\mathcal{A}_4$ wins with exactly the same probability that $\mathcal{Z}$ distinguishes between the two games, which equals the probability that Sim outputs $\perp$. This occurs only when $\mathsf{Niwi.Verify}(\mathsf{stm}, \pi) = 1$ and $\mathsf{R}_{\mathcal{U}}(\mathsf{stm}, \mathsf{wts}) \neq 1$, precisely the event that makes $\mathcal{A}_4$ win in the argument of knowledge game according to Fig. 11. Since $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ satisfies the argument of knowledge property by assumption, this probability must be negligible.

Finally, we recall that the argument of knowledge property holds also against an adversary that can concurrently use WI proof of other sessions. We directly refer to the discussion in Sec. 2 for more details on this.

Thus, we conclude that $\mathsf{Game}_{3\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{c}{\approx} \mathsf{Game}_{4\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

$\underline{\mathsf{Game}_5}$. In this game, we show that the extraction of the message m performed by Sim from the commitment $\tau_{\mathcal{U}}$ outputs $\perp$ only with negligible probability when the commitment can be successfully opened (i.e., is well-formed).

Note that when the extraction from the commitment $\tau_{\mathcal{U}}$ of the committed message m outputs $\perp$, the simulator Sim aborts. Therefore, we need to analyze the probability of such an event occurring.

First, we recall that (as stated in the previous games) the straight-line extractability property holds also against an adversary that can rely on hiding commitment generated in other concurrent sessions. We directly refer to the discussion in Sec. 2 for more details on this.

Next, we emphasize that our concern here is specifically that the $\mathsf{Com.Ext}$ algorithm as specified in Fig. 13 outputs $\perp$ only with negligible probability, rather than ensuring it outputs the message to which $\tau_{\mathcal{U}}$ commits. That is, we focus exclusively on the event where $\mathsf{Com.Ext}$ outputs $\perp$.This distinction is central to our blind signature scheme, particularly in how we define the relation $\mathsf{R}_{\mathcal{S}}$ and why our construction is not black-box with respect to a straight-line extractable and provable commitment scheme Com.

That is, specifically, as detailed in App. E, it is possible for adversary $\mathcal{A}$ to compute $\tau_{\mathcal{U}}$ such that it can never be decommitted (i.e., the commitment is bogus, still well-formed). In such cases, according to App. E, the extractor $\mathsf{Com.Ext}$ might output a message even though $\tau_{\mathcal{U}}$ cannot be properly decommitted because it is fundamentally invalid.

Our focus is therefore on a specific property of our commitment scheme Com. We analyze only the case where the extractor outputs $\perp$, not cases where extraction "succeeds" (not output $\perp$) but yields incorrect results (because the commitment is bogus). This distinction is crucial since extraction might technically "succeed" (not output $\perp$) even when $\tau_{\mathcal{U}}$ suffers from over-extraction. We accept over-extractable commitments because no proof of correct opening can later be computed with them. We define the event where $\mathsf{Com.Ext}$ outputs $\perp$ as $\mathsf{E}^{\tau_{\mathcal{U}}}_{\mathsf{Ext}}$.

We now show that $\mathsf{E}^{\tau_{\mathcal{U}}}_{\mathsf{Ext}}$ occurs only with negligible probability. First, we note that before performing extraction, the simulator Sim verifies that commitment $\tau_{\mathcal{U}}$ is well-formed. If not well-formed, Sim terminates the interaction without responding to adversary $\mathcal{A}$, mirroring the behavior of an honest signer $\mathcal{S}$. Only when this check passes, Sim obtains from the $\mathcal{G}_{\mathsf{oRO}}$ functionality the queries made by $\mathcal{A}$ in the current session (and any illegitimate queries from $\mathcal{Z}$), and then calls $\mathsf{Com.Ext}$.

By the analysis in the proof of Thm. 2, it is immediate to conclude that the event $\mathsf{E}^{\tau_{\mathcal{U}}}_{\mathsf{Ext}}$ occurs only with negligible probability. Informally, this follows from considering the probability that $\mathcal{A}$ makes exactly one RO query for each $\mathsf{a}_1, \ldots, \mathsf{a}_r$, in $\tau_{\mathcal{U}}$, when generating a valid commitment $\tau_{\mathcal{U}}$. Specifically, the analysis accounts for both the number of $\mathsf{a}_1, \ldots, \mathsf{a}_r$ in $\tau_{\mathcal{U}}$ and the complexity of the "proof of work" (i.e., the requirement that the RO output contains a specified number of zero bits). For a detailed argument, we refer directly to Thm. 2, as it applies unchanged in this setting.

$\underline{\mathsf{Game}_6}$. We now address the distinction between verification through list comparisons (as done in the ideal functionality) and verification through the `Verify` algorithm (as specified in Fig. 8). In Game_6, whenever a party $\mathcal{P}$ invokes the `Verify` procedure with input $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$, we exclusively employ algorithmic verification via the `Verify` algorithm in Fig. 8 rather than consulting List_{σ}. Specifically, verification now always proceeds as follows: (1) Parse σ as π for $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$, (2) Compute $b = \mathsf{Niwi.Verify}(\mathsf{stm}, \pi)$, where the statement $\mathsf{stm} = (\mathsf{pk}, \mathsf{m})$, and (3) Output b as the verification result.[27]

We now prove that Game_5 is computationally indistinguishable from Game_6, based on the unforgeability property of the signature scheme Sig described in Def. 8.

Consider the event in Game_5 where some party $\mathcal{P}$ requests verification of a signature via $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$ such that no entry $(\mathsf{m}, *, \mathsf{pk}, 1)$ exists in List_{σ}, yet the `Verify` algorithm returns 1. This would enable the environment $\mathcal{Z}$ to distinguish between Game_5 and Game_6[28].

[27] Note that in Game_5, this strategy was used only when $(\mathsf{m}, *, \mathsf{pk}, 1) \notin \mathsf{List}_{\sigma}$ and the signer $\mathcal{S}$ was honest, or when $(\mathsf{m}, \sigma, \mathsf{pk}, *) \notin \mathsf{List}_{\sigma}$; otherwise the output was taken directly from List_{σ}. In contrast, in Game_6 this strategy is applied universally.

[28] Note that the reverse scenario, where `Verify` outputs 0 but an entry $(\mathsf{m}, *, \mathsf{pk}, 1)$ exists in List_{σ}, cannot occur due to the perfect completeness of the Niwi scheme.

Assume, by contradiction, that there exists an environment $\mathcal{Z}$ that distinguishes between Game_5 and Game_6 with non-negligible probability. We construct an adversary $\mathcal{A}_6$ against the unforgeability of Sig as follows. $\mathcal{A}_6$ receives a public key pk' from the unforgeability challenger for the signature scheme Sig. $\mathcal{A}_6$ computes the public key pk by embedding the challenger's public key pk', and computing a commitment $\tau_{\mathcal{S}}$ to the message 0 as in Game_5, setting $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$. $\mathcal{A}_6$ internally emulates the execution of Game_5 with one key modification: when computing a signature σ' on message $\tau_{\mathcal{U}}$, $\mathcal{A}_6$ queries the challenger's signing oracle Sign with input $\tau_{\mathcal{U}}$. $\mathcal{A}_6$ then continues the simulation as in Game_5, namely by computing π' using the $\mathsf{Niwi.Prove}$ algorithm and sending (σ', π'). $\mathcal{A}_6$ maintains a list $\mathsf{List}_{\mathcal{A}_6}$, initially empty. Whenever a message $\tau_{\mathcal{U}}$ is sent from a (corrupted) user $\mathcal{U}$, $\mathcal{A}_6$ extracts as in Game_5 the possible message m[29] committed in $\tau_{\mathcal{U}}$. Note that the extraction procedure $\mathsf{Com.Ext}$ outputs $\perp$ only with negligible probability, as established in the analysis of Game_5, and this remains an invariant in the current setting. Then, $\mathcal{A}_6$ computes σ', and before sending (σ', π'), stores m in $\mathsf{List}_{\mathcal{A}_6}$. When a party $\mathcal{P}$ sends a request $(\mathsf{VerifyQuery}, \mathsf{sid}, \mathsf{m}, \sigma, \mathsf{pk})$ for a signature verification, $\mathcal{A}_6$ performs the following checks:

1. Verify that $\mathsf{m} \notin \mathsf{List}_{\mathcal{A}_6}$.
2. Parse σ as π for $\mathsf{Niwi}[\mathsf{R}_{\mathcal{U}}]$ and check if the `Verify` algorithm (i.e., the NIWI verification of π with statement $(\mathsf{pk}, \mathsf{m})$) accepts.

If either check fails, $\mathcal{A}_6$ continues without action. If both checks succeed, $\mathcal{A}_6$ leverages the witness wts extracted by $\mathsf{Niwi.Ext}$ (according to Game_5) from the signature σ (i.e., the proof π). That is, $\mathcal{A}_6$ obtains the witness $(\mathsf{d}_{\mathcal{U}}, \sigma', \mathsf{sk}, \tau_{\mathcal{U}})$.

Finally, $\mathcal{A}_6$ checks if it has extracted sk such that it is equal to (sk', r), and such that by computing $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{Sig.KG}(1^\lambda; r)$ the public key pk' is (part of) the public key pk (i.e., $\mathcal{A}_6$ checks if obtains the trapdoor witness has been used to compute the signature σ). If so, $\mathcal{A}_6$ picks a random message m^* such that $\mathsf{m}^* \notin \mathsf{List}_{\mathcal{A}_6}$, $\mathcal{A}_6$ runs $\sigma^* \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}', \mathsf{m}^*)$ and return the tuple (m^*, σ^*).

If not, returns the tuple $(\tau_{\mathcal{U}}, \sigma')$ to the unforgeability challenger. We note that $\tau_{\mathcal{U}}$ was never signed by the oracle Sign; otherwise, the corresponding message m would have appeared in $\mathsf{List}_{\mathcal{A}_6}$ due to the extraction from $\tau_{\mathcal{U}}$ performed by Sim.

Thus, we have successfully reduced the distinguishing advantage of $\mathcal{Z}$ between Game_5 and Game_6 to the unforgeability of the signature scheme Sig. By assumption, forgery attempts against Sig succeed only with negligible probability. Therefore, we conclude that $\mathsf{Game}_{5\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{c}{\approx} \mathsf{Game}_{6\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

$\underline{\mathsf{Game}_7}$. In this game, we modify Game_6 by changing how the commitment $\tau_{\mathcal{S}}$ is computed during the key generation step by Sim. Specifically, while in Game_6 the commitment $\tau_{\mathcal{S}}$ (which is part of the public key pk) is a commitment to the message 0 using the $\mathsf{Com.Sen}$ algorithm, in Game_7 the commitment $\tau_{\mathcal{S}}$ is computed by committing to the message $\mathsf{sk}'\|r$ using the algorithm $\mathsf{Com.Sen}$, with Sim storing the associated decommitment information $\mathsf{d}_{\mathcal{S}}$. After this change, the two games remain identical.

We now prove that Game_6 is computationally indistinguishable from Game_7, based on the hiding property of the commitment scheme Com described in Def. 3. Assume by contradiction that there exists an environment $\mathcal{Z}$ that distinguishes between Game_6 and Game_7 with non-negligible probability. We construct an adversary $\mathcal{A}_7$ against the hiding property of Com as follows. $\mathcal{A}_7$ receives the public parameters for the commitment scheme from the hiding experiment challenger. $\mathcal{A}_7$ internally emulates the entire execution of Game_6. When Sim needs to create the commitment $\tau_{\mathcal{S}}$, $\mathcal{A}_7$ submits the pair $(0, \mathsf{sk}'\|r)$ to the hiding experiment challenger. Upon receiving a commitment from the challenger, $\mathcal{A}_7$ uses it as $\tau_{\mathcal{S}}$. $\mathcal{A}_7$ continues the simulation until $\mathsf{Niwi.Ext}$ is executed (during the verification phase on input the proof π, namely the signature σ) and $\mathcal{A}_7$ obtains a witness wts as in Game_6. $\mathcal{A}_7$ checks the witness $\mathsf{wts} = (\mathsf{d}_{\mathcal{U}}, \sigma, \mathsf{sk}, \tau_{\mathcal{U}})$ and specifically it parses sk as $(\mathsf{sk}^*, r^*, \mathsf{d}^*_{\mathcal{S}})$ and checks if $\mathsf{sk}'\|r$ equals to $\mathsf{sk}^*\|r^*$ if it is the case outputs 1. If this is not the case, $\mathcal{A}_7$ continues the simulation until $\mathcal{Z}$ produces its output, at which point $\mathcal{A}_7$ outputs the same guess.

Observe that when the challenger commits to 0, $\mathcal{A}_7$ perfectly simulates Game_6; when it commits to $\mathsf{sk}'\|r$, $\mathcal{A}_7$ perfectly simulates Game_7. That is, after extracting the trapdoor witness, $\mathcal{A}_7$ directly breaks the hiding property of Com in case the trapdoor witness was used. Otherwise, $\mathcal{A}_7$ distinguishes between the two games with exactly the same probability as $\mathcal{Z}$. Since Com satisfies the hiding property (as proved in Thm. 1), that $\mathcal{A}_7$ wins must be negligible.

Finally, we note that according to the discussion in Sec. 2, we presented a definition of hiding (see Def. 3) that enables security also against malleability attacks. In particular, we prove that the

[29] We remark that this is the "possibly committed" message because Com, as described in App. E, is subject to over-extraction. Thus, while this commitment might not be correctly decommittable, if it can be decommitted, m is the only possible message.

hiding of the commitment issued in a given session is maintained, even if the extractor is executed concurrently in different sessions.

Thus, we conclude that $\mathsf{Game}_{6\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathsf{Game}_{7\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

Game_8. In this game, building upon Game_7, we modify the simulator Sim's strategy further. Specifically, in Step 5 of signature generation when $\mathcal{U}$ is corrupted, Sim still runs the algorithm $\mathsf{Niwi.Prove}$ for the relation $\mathsf{R}_{\mathcal{S}}$ using statement $(\tau_{\mathcal{U}}, \mathsf{pk})$ where $\mathsf{pk} = (\mathsf{pk}', \tau_{\mathcal{S}})$, but we change the witness used to produce π'.

Instead of using the trapdoor witness $(s_A, s_B, \mathsf{d}_A, \mathsf{d}_B)$ extracted from $\tau_{\mathcal{U}}$, we now use the witness that an honest signer $\mathcal{S}$ would use, following the description in Fig. 8. That is, Sim runs $\mathsf{Niwi.Prove}$ for relation $\mathsf{R}_{\mathcal{U}}$ on input statement $(\tau_{\mathcal{U}}, \mathsf{pk})$ with witness sk equals to $(\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$[30].

We now prove that Game_7 is computationally indistinguishable from Game_8 based on the WI property of the NIWI scheme $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ described in Def. 10. Assume by contradiction that there exists an environment $\mathcal{Z}$ that distinguishes between Game_7 and Game_8 with non-negligible probability. We construct an adversary $\mathcal{A}_8$ against the WI property of $\mathsf{Niwi}[\mathsf{R}_{\mathcal{S}}]$ as follows. $\mathcal{A}_8$ receives the public parameters for the NIWI scheme from the WI experiment challenger. $\mathcal{A}_8$ then internally emulates the entire execution of Game_7. When generating the proof π, $\mathcal{A}_8$ prepares: (a) the trapdoor witness wts_0, that is the tuple $(s_A, s_B, \mathsf{d}_A, \mathsf{d}_B)$, and (b) the honest witness wts_1, that is $\mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$. $\mathcal{A}_8$ then submits the statement $\mathsf{stm} = (\tau_{\mathcal{U}}, \mathsf{pk})$ along with both witnesses wts_0 and wts_1 to the WI challenger's proving oracle Prove. Upon receiving the proof π', $\mathcal{A}_8$ uses it as the NIWI that is part of the message sent by Sim to $\mathcal{A}$ along with σ'. $\mathcal{A}_8$ continues the simulation until $\mathcal{Z}$ produces its output, at which point $\mathcal{A}_8$ outputs the same guess.

Note that both witnesses satisfy the relation $\mathsf{R}_{\mathcal{U}}$, and therefore the proving oracle Prove outputs $\perp$ only with negligible probability. The witness $\mathsf{wts}_0 = (s_A, s_B, \mathsf{d}_A, \mathsf{d}_B)$ satisfies the relation via the trapdoor branch, except in the negligible case where extraction from $\tau_{\mathcal{U}}$ fails and outputs $\perp$. This property remains invariant: whenever extraction fails, a direct reduction to the extractability property of the commitment scheme, as shown in Game_5, can be constructed, thereby ensuring that the probability of the extractor $\mathsf{Com.Ext}$ outputting $\perp$ is negligible. Note that we do not consider here the possibility that the commitment $\tau_{\mathcal{U}}$ is not well-formed (i.e., that it may allow over-extraction of messages; see App. E), since such cases are already handled by the definition of the relation $\mathsf{R}_{\mathcal{S}}$ (see Fig. 7). The witness $\mathsf{wts}_1 = \mathsf{sk} = (\mathsf{sk}', r, \mathsf{d}_{\mathcal{S}})$ satisfies the relation as constructed by Sim.

When the WI challenger's bit equals 0, using witness wts_0, the adversary $\mathcal{A}_8$ perfectly simulates Game_7; when it equals 1, using witness wts_1, $\mathcal{A}_8$ perfectly simulates Game_8. Therefore, $\mathcal{A}_8$ breaks the WI property of Niwi with exactly the same probability that $\mathcal{Z}$ distinguishes between the two games. Since Niwi satisfies the WI property, this probability must be negligible.

Finally, we note that according to the discussion in Sec. 2, we presented a definition of WI (see Def. 3) that enables security also against malleability attacks. In particular, we prove that the WI of a proof issued in a given session is maintained, even if the extractor is executed concurrently in different sessions.

Thus, we conclude that $\mathsf{Game}_{7\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{c}}{\approx} \mathsf{Game}_{8\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}}$.

Differences with GEXEC. We can now conclude the proof. The simulator Sim behaves almost exactly like the real adversary $\mathcal{A}$ except for one step. The simulator Sim extracts the committed messages using the $\mathsf{Com.Ext}$ algorithm of Com described in Fig. 13. By making the very same arguments we have used both in Game_2 and Game_5, we can show it is possible to avoid such extraction, and in such cases, because the extraction fails only with negligible probability, a possible change between the games is noticeable only with negligible probability. We directly refer to the two games since the same idea can be applied here. Thus we conclude that $\mathsf{Game}_{8\,\mathcal{F}_{\mathsf{BS}},\mathsf{Sim},\mathcal{Z}} \overset{\mathrm{s}}{\approx} \mathrm{GEXEC}_{\phi_{\mathsf{BS}},\mathcal{A},\mathcal{Z}}$.

Combining all of the above, the output of $\mathcal{Z}$ with $\mathcal{A}$ after an execution of ϕ_{BS} in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model is computationally indistinguishable from its output after an execution with Sim and $\mathcal{F}_{\mathsf{BS}}$ in the ideal model. Therefore, protocol ϕ_{BS} GUC-realizes the ideal functionality $\mathcal{F}_{\mathsf{BS}}$ in the $\mathcal{G}_{\mathsf{oRO}}$-hybrid model. □

J Extension of ϕ_{BS} to Partially Blind Signatures

Partially Blind Signatures, first introduced in [AF96], extend the standard blind signature concept by allowing a message to consist of a public part (visible to the signer) and a secret part (hidden from the

[30] Note that the witness for relation $\mathsf{R}_{\mathcal{S}}$ is a tuple of five values where the secret key sk is one of them. Without loss of generality, we omit discussing the other four values since they can be the same as in Game_7 or length-matching values, which makes no difference.

signer). This notion has been extensively explored, as it enables real-world applications that cannot be achieved using standard blind signatures. Following the very same strategy in [Fis06a], we discuss how our protocol ϕ_{BS} can be naturally extended to support partially blind signatures.

That is, in the first round of interaction, the user $\mathcal{U}$ sends both the commitment $\tau_{\mathcal{U}}$ and a public message $\mathsf{m}_{\mathsf{pub}}$ to the signer $\mathcal{S}$. The signer then signs the concatenated value $(\tau_{\mathcal{U}} \| \mathsf{m}_{\mathsf{pub}})$, and the user $\mathcal{U}$ completes the protocol to obtain the final signature σ equals to π. The security proof follows easily from our proof of the original protocol in Thm. 3.

K Performance Analysis

Our work makes a significant theoretical contribution by demonstrating the existence of GUC-secure blind signatures assuming only OWFs in the NPRO model (i.e., by considering the global $\mathcal{G}_{\mathsf{oRO}}$ functionality in Fig. 1). To the best of our knowledge, this is the first construction achieving GUC security for blind signatures, and notably, the only (GUC-secure) protocol for blind signatures in Minicrypt.

While our analysis is primarily asymptotic and we do not claim to achieve a fully practical construction at this stage, our approach shows promising potential for real-world applications. In this section, we analyze the asymptotic performance of our construction when instantiated with CRHFs.

It is worth highlighting that while our current security analysis is conducted in the classical setting, we believe that extending it to the quantum setting is viable. Specifically, by adapting our proofs to the quantum RO model (QROM), the underlying hardness assumption of CRHFs would likely transfer well to the post-quantum regime. This is particularly promising given CRHF-based signature schemes such as SPHINCS+, which has been standardized by NIST as a post-quantum signature candidate.

We take a modular approach, first analyzing the complexity of individual cryptographic primitives before deriving the overall complexity of our blind signature scheme. We set the security parameter to $\lambda = 128$ bits throughout our analysis.

We analyze the efficiency of our protocol leveraging the following instantiation of the underlying used primitives:

1. The non-interactive straight-line extractable and provable commitment scheme presented in Fig. 5 uses the Halevi-Micali commitment [HM96] as the underlying commitment with SHA256 as the underlying CRHF (see [KLP22, Appendix A.3] for details on the commitment and its instantiation through SHA256), very similar to our use of Naor's commitment in Cor. 3[31].
2. For the signature scheme, we employ the NIST-standardized SPHINCS+ [HBD+20], which relies only on CRHFs, again using SHA256.
3. For the NIWI proofs, several efficient constructions based on CRHFs are available, including [GMO16, KKW18, BBHR19, DOT21, AHIV23, BBD+23]. In the following, we discuss which construction best suits our specific requirements and provide details on our instantiation.

SPHINCS+ signature scheme. Based on the NIST standardization reports [HWG+22, Table 2], SPHINCS+ with a security parameter $\lambda = 128$ and using SHA256 (i.e., sphincs-sha2-128f) requires 4,817 SHA256 calls for key generation, 112,937 SHA256 calls for signing, and 13,117 SHA256 calls for verification.

Extractable commitment scheme from Halevi-Micali commitment scheme. We now analyze the commitment complexity in terms of the number of SHA256 calls. Specifically, we look at the size of the scheme described in Fig. 5 when instantiated with the Halevi-Micali CRHF-based commitment [HM96] using SHA256.

We first briefly recall how the Halevi-Micali commitment works and its size and complexity in terms of SHA256 calls, then we focus on its usage in our commitment scheme. The discussion mainly follows the one in [KLP22, Appendix A.3].

Analysis of the Halevi-Micali commitment. Let η be the output length of a pair-wise independent hash family $\mathcal{H}$, and define $N{:=}4\lambda + 2\eta + 4$. The Halevi-Micali commitment for bounded message size[32]

[31] As with Naor's commitment, the first round of the Halevi-Micali commitment can be reused, and thus the same discussion from Cor. 3 applies here as well.

[32] Although our extractable commitment scheme technically requires the Halevi-Micali commitment to accept messages of variable length, we can easily handle this by first applying a CRHF with unbounded input to the message (e.g., SHA256). This allows us to work with fixed-size inputs at the cost of just one additional hash computation, which asymptotically does not affect our complexity analysis.

uses both a CRHF $H : \{0,1\}^N \to \{0,1\}^\lambda$ and a pair-wise independent hash family $\mathcal{H}$ with hash functions from $\{0,1\}^N$ to $\{0,1\}^\eta$. By using the standard construction, each element in $\mathcal{H}$ can be described using $(N + 2\lambda)$ bits.

To commit to a message m the algorithm: (1) samples a uniform string $r \in \{0,1\}^N$ and a function $h \in \mathcal{H}$ such that $h(r) = H(\mathsf{m})$, (2) computes $y = H(r)$, and (3) outputs the commitment τ as the tuple (h, y), i.e., the description of the function h from the family $\mathcal{H}$ and the output of $H(r)$. To verify the commitment, given the message m, the commitment τ equals to (h, y), and the decommitment information d equals to r, check that $h(r) = H(\mathsf{m})$ and $H(r) = y$. If both checks pass, output 1; otherwise, output 0.

Following the analysis in [KLP22], the computational complexity involves two evaluations of H for both the commitment and decommitment phases.

The commitment size consists of λ bits for the output of H and $(N + 2\lambda)$ bits for the description of $h \in \mathcal{H}$. Thus, when instantiated with SHA256, the total commitment size is $3\lambda + N$ bits, concretely 1400 bits. The decommitment consists of an N-bit string, equal to 1000 bits.

Analysis of the extractable commitment. According to Sec. 4.1, we have already computed the commitment size in bits, parameterized by the underlying commitment and decommitment information size (which in our case is the Halevi-Micali commitment).

We recall from Sec. 4.1 that the size in bits of the commitment is $r\theta(2C + D + 1)$, where C is the size of the underlying commitment, D is the size of the underlying decommitment information, and in our case, these refer to the Halevi-Micali commitment. According to our analysis, we have set $\theta = \lceil \log \lambda \rceil + \ell$ and $\ell = \lambda/r$, thus the commitment size is approximately $(r \log \lambda + \lambda)(2C + D + 1)$. By setting $r = 4$ as a reasonable value (i.e., $\ell = 32$), we obtain that the total size of the commitment is approximately 55 kilobytes. The computational complexity in terms of SHA256 calls is determined by the number of Halevi-Micali commitments required, which is $2 \cdot (r \log \lambda + \lambda)$. Since each Halevi-Micali commitment requires 2 SHA256 calls, we need approximately 650 SHA256 evaluations. For the decommitment algorithm, the operation is similar; we need to decommit to half of the Halevi-Micali commitments present in the extractable commitment, resulting in approximately 325 SHA256 evaluations.

NIWIs from efficient NIZK schemes. Based on our previous discussion, there exists a rich collection of (non-interactive) proof systems based on CRHFs that are ZK and thus are suitable for our purposes. That is, we can leverage efficient ZK schemes, some of which are also succinct, and use the well-established result that ZK implies WI [FS90].

While this approach appears straightforward, an important consideration arises. NIZK schemes in the NPRO model are impossible for non-trivial languages [Pas04], thus none of the above candidates directly fits our purpose of basing our result on the milder assumption of the NPRO model. We explain now how this limitation can be addressed in several ways for our interactive blind signature protocol. First, we observe that in our simulator strategy described in the proof of Thm. 3, even when using such NIZK schemes, the simulator does not need to program the RO. This is because we designed a strategy where the simulator can extract a trapdoor witness from the adversary to compute the necessary proof. Note that this is possible because we are building an interactive protocol that leverages the NIZK scheme, and thus we overcome the necessity of programming by having the simulator perform specific operations in the previous round of interaction. Indeed, if we aimed to realize a NIWI from only a NIZK, this would not be possible since, without interaction, there is no opportunity for the simulator to obtain such a trapdoor witness. The programmability of the RO may only be required in the reduction, which is entirely feasible at the price of giving up the pure NPRO model.

We discuss here the second approach, which we prefer. An alternative is to begin with the interactive versions of these proof systems, since most NIZK schemes in the RO are constructed modularly by first developing an interactive honest-verifier zero-knowledge (HVZK) proof system and then applying the FS transform. That is, the idea is to look if the interactive version satisfies the WI property and then convert it to a non-interactive scheme in the NPRO model by leveraging the classical result in [YZ06, Claim 1].

We propose as explicit examples two concrete paths for instantiations:

1. We can leverage the (variant of) Ligero [AHIV23] presented in [KLP22, Protocol 12]. Specifically, the authors of [KLP22, Lemma 7] establish that a (slightly) variant of Ligero achieves statistical WI. Based on this result, we can compile this protocol using the FS transform to obtain an efficient NIWI with performances comparable to Ligero.

2. We can use the KKW proof system [KKW18], a 3-round HVZK protocol. By applying the OR composition of KKW with itself to achieve WI, and then leveraging [YZ06, Claim 1], we can construct an efficient NIWI with performance comparable to two executions of KKW[33].

Note that, while we present these specific examples, our goal here is to demonstrate that constructing efficient and practically viable NIWIs from CRHFs in the NPRO model is readily achievable and (implicitly) already present in the literature.

Analysis of the NIWI. To present an asymptotic analysis of our scheme's performance, following the approach taken in recent works on potentially practical PQ round-optimal blind signatures [AKSY22, dPK22, BLNS23], we focus primarily on the signature size produced by the blind signature scheme, which includes the size of the NIWI proof for relation $\mathsf{R}_{\mathcal{U}}$ (see Fig. 7). That is, we focus on the proof size for the relation $\mathsf{R}_{\mathcal{U}}$. As discussed earlier, while several proof systems could be used, we base our analysis on Ligero [AHIV23] and specifically on the Ligero++ [BFH+20] experimental evaluation that is tight for SHA256. That is, we estimate the proof size according to [BFH+20, Figure 1], which gives experimental results that are proportional to the number of SHA256 calls one wants to prove.

Thus, it remains to analyze the number of SHA256 calls in the relation $\mathsf{R}_{\mathcal{U}}$.

That is, the circuit must compute: (1) the execution of the decommitment algorithm for the extractable commitment scheme Com, and (2) the verification of a SPHINCS+ signature. For Com scheme, each decommitment requires approximately 325 SHA256 evaluations (since we decommit only half of the total Halevi-Micali commitments in our extractable commitment scheme). The SPHINCS+ signature verification, according to the NIST report, requires an additional 13117 SHA256 evaluations. This gives us a total of approximately 13500 SHA256 evaluations, that is, less than 2^{14} calls to SHA256, which, according to [BFH+20, Figure 1] outputs a proof of approximately 250 kilobytes.

L Commitments with Extractable Proofs are not Enough

At first sight, one might argue that our commitment could be replaced with a plain-model commitment scheme together with a non-interactive straight-line extractable proof proving knowledge of the opening. This would enable straight-line extraction of the committed message from the associated proof, while provability would follow from the fact that the commitment is in the plain model. Although this approach seems promising, in reality, it is not if one wishes to achieve our lower bounds in terms of both computational and setup assumptions (i.e., OWFs and NPRO).

First, we make the following observation: we need some form of protection for the committed message, and thus we would ideally like the straight-line extractable proof to also be ZK. Unfortunately, since we only assume observability of the RO, we cannot achieve ZK; nevertheless, we can at least obtain WI. This leads to the idea of sending a plain-model commitment together with a WI straight-line extractable proof. However, we recall that: (a) for the WI property to be meaningful, it must be defined for a relation of an NP-language that admits at least two valid witnesses for the same statement, and (b) to prove hiding of the commitment, we must be able to "simulate" such a proof. In the case of a NIZK, this is straightforward since NIZKs come equipped with their own simulator that (e.g., by programming the RO) makes the reduction work smoothly. In contrast, when using a NIWI, we must address this issue differently. A natural approach (which is also what we do when computing the final signature) is to define a statement that is the disjunction of the original statement (i.e., the one used by the honest prover) and a trapdoor statement that can be used to "simulate" the NIWI[34].

Now consider how our construction works, specifically according to the intuition given in the technical overview. When the user is corrupted, the simulator needs to send to the real-world adversary (before extracting the message m) a commitment and a NIWI straight-line extractable proof of the secret key sk. To prove security, it is necessary in a sequence of games to show that the user's behavior is indistinguishable when we change the committed message and no longer commit to sk (as part of the pk). Accordingly, it would also be necessary to "simulate" the NIWI by making use of the trapdoor statement. Here, the problem arises: at the point when the simulator sends pk (containing the NIWI and the plain-model commitment), it has had no previous interaction with the (malicious) user, and moreover, it can only

[33] While the OR composition technique to achieve WI is typically established for Σ-protocols with the special-soundness property (which KKW does not satisfy), we can follow the approach in [KLP22] for Ligero, where similar WI properties were derived despite lacking standard special-soundness.

[34] Here we use the term "simulate" to indicate the ability, during the reduction, to switch to the witness for the trapdoor statement that carries no information about the commitment.

observe the queries to the RO made by the adversary, which does not help in simulating the NIWI. Thus, it is unclear how the NIWI should be computed. If we use as a trapdoor statement, say, the output of the RO on some OWF, with the witness being the input, then because the simulator cannot program the RO, it cannot program the output such that it later has a witness for the trapdoor statement. Another option is to use information provided by the malicious user, but the malicious user has not spoken yet, since the first message is the public key sent by the simulator, so there is no interaction that the simulator can leverage to obtain a witness for the trapdoor theorem.

Finally, one might hope that because the commitment itself is statistically hiding, there is more than one valid message that could have been committed and thus there are multiple witnesses without involving any trapdoor theorem. However, (a) we want a construction from OWFs (with the opening not involving a RO, due to provability), but classical lower bounds arise that prevent basing the construction on OWFs [HHRS07], and (b) in any case it is unclear how the simulator can equivocate without programming the RO or having prior interaction.

Note that our construction does not suffer from this issue, and according to the description of our commitment, we can prove its hiding property, allowing our security proof to proceed smoothly. The key insight, following our earlier presentation of the commitment scheme, is that we employ a different strategy for the security proof that relies on how the PW protocol works. Specifically, we leverage the fact that PW suffers from over-extraction [GLOV12]. This over-extraction is crucial, and we exploit it to prove the security of our construction. We provide further details in the technical section of the paper, where we formally prove the hiding property of our commitment.