Round-Optimal Pairing-Free Blind Signatures

Julia Kastner* Stefano Tessaro† Greg Zaverucha‡

January 20, 2026

Abstract

We present the first practical, round-optimal blind signatures in pairing-free groups. We build on the Fischlin paradigm (EUROCRYPT 2007) where a first signature is computed on a commitment to the message and the final signature is a zero-knowledge proof of the first signature. We use the Nyberg-Rueppel signature scheme as the basis (CCS 1993) , it is a well-studied scheme with a verification equation that is sufficiently algebraic to allow efficient proofs, that do not need to make non-black box use of a random oracle. Our construction offers flexibility for trade-offs between underlying assumptions and supports issuance of signatures on vectors of attributes making it suitable for use in anonymous credential systems. As a building block, we show how existing NIZKs can be modified to allow for straight-line extraction. We implement variants of our construction to demonstrate its practicality, varying the choice of elliptic curve and the proof system used to compute the NIZK. With conservative parameters (NIST-P256 and SHA-256) and targeting short proofs, signatures are 1 349 bytes long, and on a typical laptop can be generated in under 500ms and verified in under 100ms.

1 Introduction

Blind signatures [Cha82] allow a signer to sign a message for the user in such a way that the message-signature pair cannot be linked to the signing session (in particular, the signer does not learn the message or signature during the signing procedure). They are key enablers of privacy-preserving authentication solutions, and yield protocols for electronic cash [Cha82, CFN90, OO92] and voting [FOO93]. Blind signatures have also recently been the focus of widespread industry adoption, for instance in proposals for privacy-preserving ad-click measurement such as Apple PCM [WJCT21], in Apple's iCloud Private Relay [App21], Google One's VPN Service [Goo], and are also central to several proposals for anonymous tokens [HIP+21, Tru25].

Pairing-free Blind Signatures. The goal of this paper is to design blind signatures that (solely) rely on standard, *pairing-free*, elliptic curves (such as NIST-P256 or Ed25519)—we refer to them generically as "pairing-free blind signatures."

Pairing-free curves remain heavily preferred for applications not expected to withstand post-quantum attacks (typically, as in the case of blind signatures, because post-quantum solutions are expensive, and privacy can be ensured unconditionally). As a result, they are widely used by standards and libraries. A major appeal are well-established and *stable* quantitative hardness

*Centrum Wiskunde & Informatica, Amsterdam, The Netherlands, julia.kastner@cwi.nl
†University of Washington, Seattle, USA, tessaro@cs.washington.edu
‡Microsoft Research, Redmond, USA, gregz@microsoft.com

assumptions for a given curve size—often, the best attacks are the generic ones. In contrast, pairing-friendly curves, while enabling richer functionalities and easier solutions (such as very simple blind signatures [Bol03]), are subject to non-generic finite-field discrete logarithm attacks in the target group. This makes their security less certain, especially in light of relatively recent algorithmic progress [KB16], when compared to the stable security of pairing-free curves. Pairing-friendly curves are also not often standardized, and are harder to instantiate than their pairing-free counterparts.

The problem of designing paring-free blind signatures has a long history, and finding secure and efficient solutions has been a challenging problem. For example, proofs of security for blind Okamoto-Schnorr signatures [Oka94] have been given [PS00, HKL19], and similarly, blind Schnorr signatures [CP93] have been proved secure [FPS20, KLX22] additionally using the Algebraic Group Model (AGM) [FKL18]. (All proofs assume the random oracle model (ROM) [BR93].) These security proofs however can only guarantee unforgeability against attackers that can only interact with the signer in a bounded number of sessions, and this restriction is not a theoretical limitation–the recently uncovered ROS attack [BLL+21] very efficiently compromises the security of these schemes given sufficiently many sessions.

Only recently, a number of schemes have been proved secure in a fully concurrent setting. The most efficient [FPS20, KLX22, TZ22] are only proven secure in the AGM+ROM, whereas a few more recent proposals [CATZ24, KRW24, KR25, BHKR26] dispense with the AGM while achieving somewhat worse efficiency. Still, as we argue next, these schemes struggle to find adoption.

Round-optimal blind signatures. A significant limiting factor in all existing pairing-free blind signatures is that they are not round optimal, i.e., they require at least *three* moves of interaction. This is often undesirable, especially in Internet-facing applications, as each signing session requires the signer to maintain some small amount of state between the first and the third move. Such state makes the system prone to denial-of-service attacks, but also opens the door to the risk of security critical implementation errors. For example, in protocols such as [TZ22, CKM+23], accidental re-use of the first signer's message leads to leaking the signing key. It is also important to mention that the lack of round optimality is what makes concurrent attacks such as ROS possible–for round-optimal protocols, sequential and concurrent security are clearly equivalent.

This fact has led to a somewhat anachronistic situation where *blind RSA signatures* [BNPS03, Lys23], which are round optimal, remain the standard choice for practical deployments, specified in RFC [DJW23]. Their round-optimality plays to their advantage despite the use of RSA-based cryptography, which is often less preferable to elliptic-curve options in other contexts.

The following question hence arises naturally:

Can we design efficient, pairing-free and round optimal blind signatures?

Before moving on, we should note here that one can build a round-optimal pairing-free blind signature fairly easily by instantiating Fischlin's generic transform [Fis06] with components that only rely on pairing-free signatures and hash functions, but it is a priori not clear how to obtain a construction which is reasonably efficient.

Our contributions. In this paper, we provide an affirmative answer to the above question by providing the first efficient pairing-free blind signature.

We do build on the aforementioned Fischlin paradigm, where a first (non-blind) signature is computed on a commitment to the message and the final blind signature is a zero-knowledge proof of knowledge of this non-blind signature. Fischlin's template, when applied generically, typically

runs into two efficiency bottlenecks. The first is that the zero-knowledge proof of knowledge of a non-blind signature depends on its verification procedure, and this in turn can involve the evaluation of a hash function (such as SHA-256) on a secret value, making the cost of such a proof high due to the non-black-box dependence on this hash function. The second is that the commitment sent by the user to the signer also needs to come with a proof of validity (we refer to this as the *issuance* proof), which can also be inefficient depending on the specifics of the used commitment.

To address these efficiency bottlenecks, our instantiation leverages the specific algebraic structure of a signature scheme by Nyberg and Rueppel (NR) [NR93] and a modified variant (mNR) due to [AdM04]. A core point is that the verification equation for an NR/mNR signature is sufficiently algebraic to always allow the proof of knowledge of an NR/mNR signature not to rely on a cryptographic hash function in a non-black box manner. Furthermore, two of our schemes rely on mNR, and for these the issuance proof can also avoid non-black-box use of the hash function.

Our construction template offers flexibility for trade-offs between underlying assumptions and one variant supports issuance of signatures on vectors of attributes making it suitable for use in anonymous credential systems. An overview of our three schemes can be seen in Table 1. As a building block, we show how existing NIZKs can be modified to allow for straight-line extraction. In all cases, our proof of unforgeability assumes the security of the underying NR/mNR scheme, along with the witness extended emulation property of the involved proof systems. We can instantiate the schemes to obtain statistical blindness.

We implement variants of our construction to demonstrate its practicality, varying the choice of elliptic curve and the proof system used to compute the NIZK. With conservative parameters (NIST-P256 and SHA-256) and targeting short proofs, signatures are 1 349 bytes long, and on a typical laptop can be generated in under 500ms and verified in under 100ms.

	Type	Msg	Hash	Security	Issuance proof	Proof of sig.
Scheme_1	blind sig.	m	H^m	EUF-CMA of mNR	Σ-proof	SNARK
Scheme_2	blind sig.	m	SHA256(m)	EUF-CMA of NR	SNARK+	SNARK
Scheme_3	credentials	$\vec{m}$	$\prod {H_i}^{m_i}$	EUF-CMA of mNR	Σ-proof	SNARK

Table 1: Three variants of our construction providing different functionality, security and performance. Scheme_1 and Scheme_2 are blind signature schemes, that hash the message in different ways. In Scheme_1 the message space is $\mathbb{Z}_p$ and the message is directly encoded as a group element via exponentiation, and our security analysis relies on the security of the modified-NR scheme. In Scheme_2 we hash the message with SHA-256 (and encode to a group element), following the original definition of NR, and therefore reduce to the security of the standard scheme. Scheme_3 generalizes Scheme_1 to sign vectors of messages, as required by anonymous credential systems and allows credentials to be presented arbitrarily many times unlinkably. Scheme_2 requires proof of a SHA-256 preimage during issuance, "SNARK+" indicates that this proof is considerably more expensive than the other SNARK proofs for Π_{NR} (having about 5x more constraints).

Concurrent Work In a concurrent work titled "On the Impossibility of Round-Optimal Pairing-Free Blind Signatures in the ROM" by Marian Dietz, Julia Kastner, and Stefano Tessaro [DKT26], it is shown that round optimal blind signatures cannot be achieved in pairing-free groups with a random oracle if both the group and the random oracle are accessed in a black-box fashion. This complements our work as it demonstrates that non-black-box techniques, such as the use of a conversion function To$\mathbb{Z}_p$ or zero-knowledge proofs, are necessary for achieving round-optimal blind signatures in pairing-free groups.

1.1 Technical Overview

Starting Point: The Fischlin Construction and Its Modifications Our construction is loosely based on the Fischlin-method [Fis06] to construct blind signatures using the following building blocks:

- a EUF-CMA secure digital signature scheme Σ
- a non-interactive zero knowledge (NIZK) proof of knowledge proof system Π
- a commitment scheme C

The Fischlin scheme proceeds as follows: The signer's key pair consists of a key pair $(\mathsf{sk}, \mathsf{vk})$ of the underlying signature scheme. Signature issuance proceeds as follows. The user commits to the message using the commitment scheme C to produce a commitment com and sends it to the signer. The signer signs the commitment com, producing signature σ and sends σ back to the user. The user then produces a zero-knowledge proof of knowledge π of the signature σ on a commitment com of the message m and outputs the signature $\sigma' = \pi$.

A recent line of work [dPK22, KNR24] has modified this Fischlin-transform as follows. Instead of using a generic commitment scheme C and signature scheme Σ, use a commitment scheme that is "compatible" with Σ in the sense that the user can commit to a message m, the signer can then issue a pre-signature σ' on com and the user can "decommit" the signature σ' to a signature σ on m. For technical reasons, the user also needs to provide a zero-knowledge proof that this commitment was generated honestly, usually proving knowledge of the message and randomness used in the commitment. In the constructions in the literature, as well as in our construction, this signature σ is linkable to σ' and thus cannot be output as the final signature on the message m. However, it allows the user to produce a proof of knowledge π of a signature σ on the message m. The proof is simpler without the commitment, and our chosen signature scheme (NR) has a relatively SNARK-friendly verification equation – in particular, we do not need to include the hash function in the circuit.

We instantiate this variant of the Fischlin-transform to obtain three versions of our scheme which are depicted in Table 1.

For these instantiations, we need the following building blocks

- A signature scheme Σ in pairing-free groups.
- A compatible commitment scheme C.
- A NIZK with witness-extended emulation.[1] Π_{iss} used by the user during issuance to prove knowledge of the message m and commitment randomness R.
- A NIZK proof of knowledge Π_{NR} that the user uses to generate the final signature.

In the following, we describe how we instantiate these building blocks.

(Modified) Nyberg-Rueppel Signatures and Compatible Commitments We instantiate the signature scheme by Nyberg-Rueppel signatures.

For schemes 1 and 3 we use a modified variant of Nyberg-Rueppel signatures due to [AdM04] which we denote by Σ_{mNR}. Key generation samples a secret key $x \leftarrow_{\$} \mathbb{Z}_p$ and a group element H. The public key consists of the group generator G, $Y = G^x$ and H. To sign a scalar $m \in \mathbb{Z}_p$,

[1] Witness extended emulation is a property that implies straight-line extraction. Informally speaking, a proof scheme is witness-extended emulatable when it is indistinguishable for an adversarial prover whether he interacts with an honest verifier, or an emulator that extracts a witness and verifies its validity.

sample $k \leftarrow_\$ \mathbb{Z}_p$ and compute $R = G^{-k} \cdot H^m$ as well as $r = \text{To}\mathbb{Z}_p(R)$. Set $s = k - r \cdot x$ and output the signature (R, s). For public key $(G, H, Y) \in \mathbb{G}^3$ and conversion function $\text{To}\mathbb{Z}_p : \mathbb{G} \to \mathbb{Z}_p$, and a signature $\sigma = (R, s)$ verification algorithm checks that

$$H^m = R \cdot Y^r \cdot G^s$$

where $r = \text{To}\mathbb{Z}_p(R)$. The commitment scheme used in this case is essentially a Pedersen commitment. For Scheme_1, the commitment is of the form $H^m \cdot G^{-k_0}$ where k_0 is the commitment randomness. For Scheme_3, the commitment is of the form $G^{-k_0} \cdot \prod_{i=1}^{n} H_i^{m_i}$ where the H_i are part of the signer's public key.

For Scheme_2 we use plain Nyberg-Rueppel signatures [NR93] denoted by Σ_{NR} to sign a group element M which is derived as $\mathsf{H}_m(M)$ for a hash function H_m that we instantiate as SHA256 in the implementation. The group element M takes the role of H^m from above. For public key $(G, Y) \in \mathbb{G}^2$ and conversion function $\text{To}\mathbb{Z}_p : \mathbb{G} \to \mathbb{Z}_p$, hash function $\mathsf{H}_m : \{0,1\}^* \to \mathbb{G}$ and a signature $\sigma = (R, s)$, the verification algorithm checks that

$$\mathsf{H}_m(m) = R \cdot Y^r \cdot G^s$$

where $r = \text{To}\mathbb{Z}_p(R)$. For this variant, the commitment is computed as $\mathsf{H}_m(m) \cdot G^{-k_0}$ where k_0 is the commitment randomness.

Our Blind Signature Construction Our blind signature issuance proceeds as follows - for simplicity we explain scheme 1. First, the user samples its share k_0 of the signature randomness k. It then computes $R_0 = G^{-k_0} \cdot H^m$ and sends it to the signer along with a proof of knowledge π_{iss} of k_0 and m. The signer verifies π_{iss}, samples its share of signature randomness k_1, and sets $R = G^{-k_1} \cdot R_0 = G^{-(k_0+k_1)} \cdot H^m$. It computes $r = \text{To}\mathbb{Z}_p(R)$ and $s_1 = k_1 - rx$. It sends s_1 and R to the user who completes the signature to $(R, s = s_1 + k_0)$. As this signature would be linkable to the signing session it was created in, instead of outputting the signature directly, the user computes a proof of knowledge π_{NR} which takes the role of the final signature.

Extraction without the full AGM For the proofs Π_{iss} and Π_{NR}, to prove knowledge of the message and commitment randomness, or a signature, respectively, we make use of a novel extraction method.

Both for Π_{iss} and Π_{NR} and regardless of which scheme is used, the prover commits to the secrets (message(s) and signature values) in a second group $\mathbb{G}'$ *of the same order as* $\mathbb{G}$ using a Pedersen commitment. Looking forward, for extraction we will assume that the adversary is *only algebraic with respect to the second group* $\mathbb{G}'$ and not $\mathbb{G}$. This technique lets us avoid having to take into account all group elements that the adversary may have access to in the first group $\mathbb{G}$, making the AGM portion of our analysis much simpler. Second, since it limits the AGM usage to the NIZKs, and only for extraction, our construction has the flexibility to use NIZKs that already support extraction.

Since the group orders of $\mathbb{G}$ and $\mathbb{G}'$ are the same, proving equality of the committed values is efficient. In the case of Π_{iss} in Scheme_1 and Scheme_3, we use a simple Σ-protocol with Fiat-Shamir [FS87] to prove equivalence of the two Pedersen commitments. Extraction follows from the AGM, and we ensure perfect zero-knowledge by adding random coins to the commitment in the second group.

For Π_{NR} as well as Π_{iss} in Scheme_2 we add a NIZK proof of knowledge of the non-arithmetic parts of the statement (evaluations of H_m and $\text{To}\mathbb{Z}_p$) and employ techniques from [OKMZ25] to bind together the group-based Σ-protocol parts of the statement and the circuit-based parts. The knowledge soundness of this NIZK proof ensures that when extracting using the AGM in $\mathbb{G}'$

from the Σ-protocol part, the witness is consistent with the statement of the NIZK. We stress that we do not need to assume straight-line extraction or WEE of the NIZK in order to achieve WEE - it suffices that the NIZK is knowledge-sound. We also show that the same second group $\mathbb{G}'$ can be used for Π_{iss} and Π_{NR}.

2 Preliminaries

Notation In this work, we consider groups $\mathbb{G}$, $\mathbb{G}'$ of prime order p. We denote group elements in upper case, e.g. G, H, Y. We denote vectors as $\vec{v}$. For a vector of group elements $\vec{X} \in \mathbb{G}^n$ and a vector of scalars $\vec{y} \in \mathbb{Z}_p$ we denote by $\vec{X}^{\vec{y}} := \prod_{i=1}^{n} X_i^{y_i}$.

We denote the security parameter by λ and negligible functions by negl. We abbreviate probabilistic polynomial time by PPT.

2.1 Groups

Definition 1 (Algebraic Algorithm)**.** *We say that an algorithm $\mathcal{A}$ is* algebraic *w.r.t. a group $\mathbb{G}$ if for any group element $Y \in \mathbb{G}$ in its output, it outputs an* explanation *$\vec{z}$ such that*

$$Y = \prod_{i=1}^{n} X_i^{z_i}$$

where $X_1, \ldots, X_n \in \mathbb{G}$ are the group elements from $\mathbb{G}$ that $\mathcal{A}$ has received as inputs so far.

2.2 (Blind) Signature Schemes

Definition 2 (Signature Scheme)**.** *A* signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *consists of the following PPT algorithms:*

$\mathsf{KeyGen}(1^\lambda)$**:** *The* key generation *algorithm takes as input the security parameter in unary encoding and outputs a key pair* $(\mathsf{sk}, \mathsf{vk})$*.*

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *The* signing *algorithm takes as input the secret key and a message and outputs a signature σ.*

$\mathsf{Verify}(\mathsf{vk}, m, \sigma)$**:** *The* verification *algorithm takes as input the verification key, a message, and a signature and outputs either* 1 *indicating that the signature is valid w.r.t. the verification key and m, or* 0 *otherwise.*

Definition 3 (Correctness)**.** *We say that a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is* correct *if for all m*

$$\Pr\left[\begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow_\$ \mathsf{KeyGen} \\ \sigma \leftarrow_\$ \mathsf{Sign}(\mathsf{sk}, m)\end{array} : \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1\right] \geq 1 - \mathrm{negl}(\lambda)$$

where we say that Σ is perfectly correct *if the above probability is* 1*.*

Definition 4 (Existential Unforgeability under Chosen Message Attacks (EUF-CMA))**.** *The game* **euf-cma** *for a signature scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and an adversary $\mathcal{A}$ is defined as follows:*

Setup. *The game samples a key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow_\$ \mathsf{KeyGen}(1^\lambda)$. It outputs vk to the adversary.*

Online Phase. *The adversary gets to query a signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ which returns signatures on the input messages. The game records the messages queried in a list Q*

Output Determination. *The adversary outputs a message signature pair* m^*, σ^*. *The game outputs* 1 *iff* $m^* \notin Q$ *and* $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*) = 1$.

The advantage is defined as $\mathrm{adv}^{\mathbf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathbf{euf\text{-}cma} = 1]$. *and we say that* Σ *is existentially unforgeable under chosen message attacks if for all PPT adversaries* $\mathcal{A}$ *the advantage* $\mathrm{adv}^{\mathbf{euf\text{-}cma}}_{\Sigma,\mathcal{A}}(\lambda) \leq \mathrm{negl}(\lambda)$.

Definition 5 (Round-Optimal Blind Signature Scheme)**.** *A* round-optimal blind signature scheme $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User} = (\mathsf{User}_1, \mathsf{User}_2), \mathsf{Verify})$ *consists of the following algorithms:*

$\mathsf{KeyGen}(1^\lambda)$ *The* key generation *algorithm takes as input the security parameter in unary encoding and outputs a key pair* $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{User}(\mathsf{vk}, m)$**:** *The* user *algorithm is split into two rounds keeping state between them.*

- $\mathsf{User}_1(\mathsf{vk}, m)$**:** *The first user algorithm takes as input a verification key and a message and outputs a user message* msgU *to the signer and saves an internal state* stU.
- $\mathsf{User}_2(\mathsf{stU}, \mathsf{msgS})$**:** *The second user algorithm takes as input the internal state* stU *and a response* msgS *from the signer and outputs a signature* σ *or* $\perp$ *(in case signature generation was unsuccessful).*

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msgU})$**:** *The* signer *algorithm takes as input a secret key* sk *and a user message* msgU *and outputs a signer response* msgS.

$\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ *The* verification *algorithm takes as input the verification key, a message, and a signature and outputs either* 1 *indicating that the signature is valid w.r.t. the verification key and* m*, or* 0 *otherwise.*

Definition 6 (Correctness)**.** *We say that a blind signature scheme* $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Verify})$ *is* correct *if for all* m

$$\Pr\left[\begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \leftarrow_\$ \mathsf{KeyGen} \\ (\mathsf{msgU}, \mathsf{stU}) \leftarrow_\$ \mathsf{User}_1(\mathsf{vk}, m) \\ \mathsf{msgS} \leftarrow_\$ \mathsf{Sign}(\mathsf{sk}, \mathsf{msgU}) \\ \sigma \leftarrow_\$ \mathsf{User}_2(\mathsf{stU}, \mathsf{msgS})\end{array} : \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1\right] \geq 1 - \mathrm{negl}(\lambda)$$

where we say that BS *is* perfectly correct *if the above probability is* 1.

Definition 7 (Blindness)**.** *The game* **blind** *for a blind signature scheme* $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Verify})$ *and an adversary* $\mathcal{S}$ *as follows:*

Setup. *The adversary* $\mathcal{S}$ *outputs a verification key* vk *and two messages* m_0, m_1. *The game chooses a bit* $b \leftarrow_\$ \{0, 1\}$.

Online Phase. *The adversary gets to make a call to the oracles* $\mathsf{User}_{1,0}$ *which internally runs* $(\mathsf{msgU}_0, \mathsf{stU}_b) \leftarrow_\$ \mathsf{User}_1(\mathsf{vk}, m_b)$ *and outputs* msgU_0 *and* $\mathsf{User}_{1,1}$ *which internally runs* $(\mathsf{msgU}_1, \mathsf{stU}_{1-b}) \leftarrow_\$ \mathsf{User}_1(\mathsf{vk}, m_{1-b})$ *and outputs* msgU_1. *It is furthermore given access to oracles* $\mathsf{User}_{2,0}$ *and* $\mathsf{User}_{2,1}$ *which take as input a signer message* msgS_0 *and* msgS_1 *respectively and internally run* $\sigma_b \leftarrow_\$ \mathsf{User}_2(\mathsf{stU}_b, \mathsf{msgS}_0)$ *and* $\sigma_{1-b} \leftarrow_\$ \mathsf{User}_2(\mathsf{stU}_{1-b}, \mathsf{msgS}_1)$.

Output Determination. *Once both oracles* $\mathsf{User}_{2,0}$ *and* $\mathsf{User}_{2,1}$ *have been called, the game outputs* (m_0, σ_0) *and* (m_1, σ_1) *to the adversary iff* $\sigma_0 \neq \perp$ *and* $\sigma_1 \neq \perp$. *Otherwise it outputs* $\perp$. *The adversary outputs a bit* b'.

The advantage is defined as $\mathrm{adv}^{\mathbf{blind}}_{\mathsf{BS},\mathcal{S}}(\lambda) := 2 \cdot |\Pr[b = b'] - 1/2|$. *We say that* BS *is* computationally blind *if for all PPT adversaries* $\mathcal{S}$ *the advantage* $\mathrm{adv}^{\mathbf{blind}}_{\mathsf{BS},\mathcal{S}} \leq \mathrm{negl}(\lambda)$, *we say that* BS *is* statistically blind *if for all adversaries* $\mathcal{S}$ *the advantage* $\mathrm{adv}^{\mathbf{blind}}_{\mathsf{BS},\mathcal{S}} \leq \mathrm{negl}(\lambda)$, *and we say that* BS *is* perfectly blind *the advantage* $\mathrm{adv}^{\mathbf{blind}}_{\mathsf{BS},\mathcal{S}} = 0$.

Definition 8 (One-More Unforgeability (OMUF))**.** *The game* ℓ-**omuf** *for a blind signature scheme* $\mathsf{BS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Verify})$ *and an adversary* $\mathcal{U}$ *as follows:*

Setup. *The game samples a key pair* $(\mathsf{sk}, \mathsf{vk}) \leftarrow_{\$} \mathsf{KeyGen}(1^{\lambda})$. *It outputs* vk *to the adversary.*

Online Phase. *The adversary gets to query a signing oracle* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *which returns signer messages* msgS *up to* ℓ *times.*

Output Determination. *The adversary outputs* $\ell + 1$ *message-signature pairs* $(m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1})$. *The game outputs* 1 *if for all* $i \neq j$ $m_i \neq m_j$ *and for all* $i \in [\ell + 1]$ *it holds that* $\mathsf{Verify}(\mathsf{vk}, m_i, \sigma_i) = 1$. *Otherwise it outputs* 0.

The advantage is defined as $\mathrm{adv}^{\ell\textbf{-omuf}}_{\mathsf{BS},\mathcal{U}}(\lambda) := \Pr[\ell\textbf{-omuf} = 1]$. *and we say that* BS *is* ℓ*-one-more unforgeable if for all PPT adversaries* $\mathcal{U}$ *the advantage* $\mathrm{adv}^{\ell\textbf{-omuf}}_{\mathsf{BS},\mathcal{U}}(\lambda) \leq \mathrm{negl}(\lambda)$.

2.3 NIZKs and SNARKs

We follow the definitions from [KRW24] for non-interactive proof systems.

Definition 9 (NP-Relation and Language)**.** *Let* $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^\star$ *be a binary relation. We say that* $\mathcal{R}$ *is an* NP-relation *if there are polynomials* p *and* q *such that* $\mathcal{R}$ *can efficiently be decided and for every* $(x, w) \in \mathcal{R}$ *we have* $|x| \leq p(\lambda)$ *and* $|w| \leq q(|x|)$. *We denote by* $\mathcal{L}_{\mathcal{R}} = \{x \in \{0,1\}^* | \exists w s.t.\ (x, w) \in \mathcal{R}\}$ *the language induced by* $\mathcal{R}$.

Definition 10 (Non-Interactive Zero-Knowledge (NIZK) Proof System in the ROM)**.** *A* Non-Interactive Zero-Knowledge (NIZK) Proof System $\Pi = (\mathsf{Prove}, \mathsf{Verify})$ *for a NP-relation* $\mathcal{R}$ *consists of the following algorithms:*

- $\mathsf{Prove}^{\mathsf{H}}(x, w)$ *takes as input a statement* x *and a witness* w *with* $(x, w) \in \mathcal{R}$ *and outputs a proof* π
- $\mathsf{Verify}^{\mathsf{H}}(x, \pi)$ *takes as input a statement* x *and a proof* π *and outputs* 1 *(*Accept*) or* 0 *(*Reject*)*

Definition 11 (Correctness)**.** *Let* $\Pi = (\mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive proof system for a relation* $\mathcal{R}$. *We say that* Π *is* correct *if for all* $(x, w) \in \mathcal{R}$ *it holds that* $\pi \leftarrow_{\$} \mathsf{Prove}^{\mathsf{H}}(x, w) \colon \mathsf{Verify}^{\mathsf{H}}(x, \pi) = 1$.

We adapt the notion of witness-extended emulation [Lin01] to algebraic adversaries. Witness-extended emulation is a stronger notion than knowledge-soundness and can be useful when a knowledge-soundness is needed in a larger protocol. In this game, a malicious prover gets to interact either with a verification oracle, or with a simulator that not only verifies the proof, but also attempts to extract a witness. It should be difficult for the malicious prover to distinguish between the two scenarios. This property will be useful in our security proofs when switching from merely verifying proofs to extracting witnesses such that the reduction can use them to simulate other oracles.

Definition 12 (Witness Extended Emulation in the AGM over $\mathbb{G}$)**.** *Let* $\Pi = (\mathsf{Prove}, \mathsf{Verify})$ *be a non interactive proof system for a relation* $\mathcal{R}$*. Let* $\mathcal{E}$ *be a PPT algorithm. Let* $\mathcal{A}$ *be an oracle algorithm and let*

$$\mathsf{Real}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{Verify}}}(1^{\lambda}) \colon b = 1].$$

$$\mathsf{Emulated}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathcal{E}}}(1^{\lambda}) \colon b = 1].$$

Here, $\mathcal{A}$ *has (black-box) access to the random oracle* H *and to an oracle* $\mathcal{O}$ *which behaves as follows:*

- $\mathcal{O}_{\mathsf{Verify}}(x, \pi)$*: returns* $\mathsf{Verify}(x, \pi)$*.*
- $\mathcal{O}_{\mathcal{E}}(x, \pi)$*: If* $\mathsf{Verify}(x, w) = 1$*, run* $w \leftarrow_{\$} \mathcal{E}(\mathcal{Q}, \mathcal{Z}, x, \pi)$ *and return 1 if* $(x, w) \in \mathcal{R}$*. Otherwise return* 0*. Here,* $\mathcal{Q}$ *denotes the set of* $\mathcal{A}$*'s queries to* H *and* $\mathcal{Z}$ *denotes the set of all algebraic explanations that* $\mathcal{A}$ *has output.*

The advantage of $\mathcal{A}$ *against witness-extended emulation is* $\mathrm{adv}^{\mathbf{wee}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda) \coloneqq |\mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Emulated}_{\mathcal{A}}(\lambda)|$*. We say that* $\mathcal{E}$ *is an* emulator in the AGM over $\mathbb{G}$ *for* Π *and knowledge relation* $\mathcal{R}$ *if for every algebraic w.r.t.* $\mathbb{G}$ *PPT algorithm* $\mathcal{A}$*, the advantage* $\mathrm{adv}^{\mathbf{wee}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)$ *is negligible in* λ*. We say that* Π *is* witness-extended emulatable in the AGM over $\mathbb{G}$ *if there exists an* emulator in the AGM over $\mathbb{G}$ *for* Π*.*

Definition 13 (Witness Extended Emulation)**.** *Let* $\Pi = (\mathsf{Prove}, \mathsf{Verify})$ *be a non interactive proof system for a relation* $\mathcal{R}$*. Let* $\mathcal{E}$ *be a PPT algorithm. Let* $\mathcal{A}$ *be an oracle algorithm and let*

$$\mathsf{Real}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{Verify}}}(1^{\lambda}) \colon b = 1].$$

$$\mathsf{Emulated}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathcal{E}}}(1^{\lambda}) \colon b = 1].$$

Here, $\mathcal{A}$ *has (black-box) access to the random oracle* H *and to an oracle* $\mathcal{O}$ *which behaves as follows:*

- $\mathcal{O}_{\mathsf{Verify}}(x, \pi)$*: returns* $\mathsf{Verify}(x, \pi)$*.*
- $\mathcal{O}_{\mathcal{E}}(x, \pi)$*: If* $\mathsf{Verify}(x, w) = 1$*, run* $w \leftarrow_{\$} \mathcal{E}(\mathcal{Q}, x, \pi)$ *and return 1 if* $(x, w) \in \mathcal{R}$*. Otherwise return* 0*. Here,* $\mathcal{Q}$ *denotes the set of* $\mathcal{A}$*'s queries to* H*.*

The advantage of $\mathcal{A}$ *against witness-extended emulation is* $\mathrm{adv}^{\mathbf{wee}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda) \coloneqq |\mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Emulated}_{\mathcal{A}}(\lambda)|$*. We say that* $\mathcal{E}$ *is an* emulator *for* Π *and knowledge relation* $\mathcal{R}$ *if for every PPT algorithm* $\mathcal{A}$*, the advantage* $\mathrm{adv}^{\mathbf{wee}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)$ *is negligible in* λ*. We say that* Π *is* witness-extended emulatable *if there exists an* emulator *for* Π*.*

Definition 14 (Knowledge-Soundness)**.** *Let* Π *be a non-interactive proof system for a relation* $\mathcal{R}$*. We say that* π *is* knowledge-sound *with* knowledge error κ *if there exists an extractor* $\mathcal{E}$ *such that for any adversary* $\mathcal{P}$ *it holds that*

$$\Pr\left[\begin{array}{c} (x, \pi) \leftarrow_{\$} \mathcal{P}^{\mathsf{H}}(1^{\lambda}) \\ b \leftarrow \mathsf{Verify}^{\mathsf{H}}(1^{\lambda}, x, \pi) \\ w \leftarrow_{\$} \mathcal{E}^{\mathcal{P}}(1^{\lambda}, x) \end{array} : b = 1 \wedge (x, w) \notin \mathcal{R}\right] \leq \kappa(\lambda).$$

Definition 15 (Zero-Knowledge)**.** *Let* Π *be a non-interactive proof system for a relation* $\mathcal{R}$*. Let* Sim *be a PPT algorithm. Let* $\mathcal{A}$ *be an oracle algorithm and let*

$$\mathsf{Real}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{Prove}}}(1^{\lambda}) \colon b = 1].$$

$$\mathsf{Simulated}_{\mathcal{A}}(\lambda) \coloneqq \Pr[b \leftarrow_{\$} \mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{Sim}}}(1^{\lambda}) \colon b = 1].$$

Here, $\mathcal{A}$ *has (black-box) access to the random oracle* H *and to an oracle* $\mathcal{O}$ *which behave as follows:*

- $\mathcal{O}_{\mathsf{Prove}}(x, w)$*: returns* $\mathsf{Prove}(x, w)$ *if* $(x, w) \in \mathcal{R}$ *and* $\perp$ *otherwise.*
- $\mathcal{O}_{\mathsf{Sim}}(x, w)$*: returns* $\mathsf{Sim}(x)$ *if* $(x, w) \in \mathcal{R}$ *and* $\perp$ *otherwise.*

The advantage of $\mathcal{A}$ *against zero-knowledge is* $\mathrm{adv}^{\mathbf{zk}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda) \coloneqq |\mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Simulated}_{\mathcal{A}}(\lambda)|$*. We say that* Sim *is a* (perfect / statistical) zero-knowledge simulator *for* Π *and knowledge relation* $\mathcal{R}$ *if for every algorithm* $\mathcal{A}$*, the advantage* $\mathrm{adv}^{\mathbf{zk}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)$ *is (0/negligible in* λ*) and we say that* Sim *is a computational zero-knowledge simulator if for every PPT algorithm* $\mathcal{A}$*, the advantage* $\mathrm{adv}^{\mathbf{zk}}_{\mathcal{A},\Pi,\mathcal{R}}(\lambda)$ *is negligible in* λ*. We say that* Π *is* (perfectly / statistically / computationally) zero-knowledge *if there exists a* (perfect / statistical / computational) zero-knowledge simulator *for* Π*.*

3 Our Blind Signature Schemes

We start by describing our two main building blocks, NR signatures and NIZKs, then move to the three instantiations of our construction template.

Nyberg-Rueppel Signatures

Definition 16 (Nyberg-Rueppel Signatures [NR93])**.** *The Nyberg-Rueppel Signature* Σ_{NR} *over a group* $\mathbb{G}$ *of prime order* p *with conversion function* $\mathrm{To}\mathbb{Z}_p : \mathbb{G} \to \mathbb{Z}_p$ *and a hash function* $\mathsf{H}_m : \{0,1\}^* \to \mathbb{G}$ *is defined as follows:*

$\mathsf{KeyGen}(\mathsf{pp}_{\mathbb{G}})$ *The key generation algorithm samples* $x \leftarrow_{\$} \mathbb{Z}_p$*, and sets* $\mathsf{sk} = x$ *and* $\mathsf{vk} = (G, Y = G^x)$*. It outputs* sk, vk

$\mathsf{Sign}(\mathsf{sk}, m \in \{0,1\}^*)$ *The signing algorithm samples* $k \leftarrow_{\$} \mathbb{Z}_p$ *and computes* $R = G^{-k}\mathsf{H}_m(m)$*. It computes* $r = \mathrm{To}\mathbb{Z}_p(R)$ *and sets* $s = k - r \cdot x$*. It outputs the signature* $\sigma = (R, s)$

$\mathsf{Verify}(\mathsf{vk}, m, \sigma = (R, s))$ *The verification algorithm checks that* $\mathsf{H}_m(m) = R \cdot Y^r \cdot G^s$ *where* $r = \mathrm{To}\mathbb{Z}_p(R)$*.*

Definition 17 (Modified Nyberg-Rueppel Signatures [AdM04])**.** *The modified Nyberg-Rueppel signature scheme* Σ_{mNR} *over a group* $\mathbb{G}$ *of prime order* p *with conversion function* $\mathrm{To}\mathbb{Z}_p : \mathbb{G} \to \mathbb{Z}_p$ *is defined as follows:*

$\mathsf{KeyGen}(\mathsf{pp}_{\mathbb{G}})$ *The key generation algorithm samples* $x \leftarrow_{\$} \mathbb{Z}_p$*,* $H \leftarrow_{\$} \mathbb{G}$ *and sets* $\mathsf{sk} = x$ *and* $\mathsf{vk} = (G, H, Y = G^x)$*. It outputs* sk, vk

$\mathsf{Sign}(\mathsf{sk}, m \in \mathbb{Z}_p)$ *The signing algorithm samples* $k \leftarrow_{\$} \mathbb{Z}_p$ *and computes* $R = G^{-k}H^m$*. It computes* $r = \mathrm{To}\mathbb{Z}_p(R)$ *and sets* $s = k - r \cdot x$*. It outputs the signature* $\sigma = (R, s)$

$\mathsf{Verify}(\mathsf{vk}, m, \sigma = (R, s))$ *The verification algorithm checks that* $H^m = R \cdot Y^r \cdot G^s$ *where* $r = \mathrm{To}\mathbb{Z}_p(R)$*.*

NIZK Proofs During issuance, our schemes make use of a NIZK proof Π_{iss} that the user uses to prove knowledge of a message and and the randomness used to generate the commitment R_0 on the message. Details on how these proof systems are instantiated for the different schemes can be found in Sections 4.1 and 4.2

To prove knowledge of the final signature, our schemes make use of a NIZK proof Π_{NR} that the user uses to prove knowledge of a signature on the message m. Details on how these proof systems are instantiated can be found in Sections 4.3 and 4.4.

3.1 Scheme_1

Our blind signature scheme Scheme_1 works as follows.

Setup Samples group parameters $\mathsf{pp}_{\mathbb{G}}$ containing generator elements, the group order p, and a description of the conversion function $\mathrm{To}\mathbb{Z}_p$.

$\mathsf{KeyGen}(\mathsf{pp}_{\mathbb{G}})$**:** $x \leftarrow_{\$} \mathbb{Z}_p, Y := G^x, H \leftarrow_{\$} \mathbb{G}$. Outputs $\mathsf{sk} = x, \mathsf{vk} = (G, H, Y)$

$\mathsf{User}_1(\mathsf{vk}, m)$**:** On input $\mathsf{vk} = (G, H, Y)$, and a message $m \in \mathbb{Z}_p$, the user samples $k_0 \leftarrow_{\$} \mathbb{Z}_p$. It then computes $R_0 := H^m \cdot G^{-k_0}$. Outputs a user message R_0 to the signer, and saves an internal state $(\mathsf{vk}, m, H^m, k_0)$. The user computes a proof of knowledge π_{iss} of (m, k_0) s.t. $R_0 = H^m G^{-k_0}$ using Π_{iss}

$\mathsf{Sign}(\mathsf{sk}, R_0)$**:** On input of the secret key sk and a user message R_0 and a proof π_{iss}, the signer verifies π_{iss} and aborts if verification fails, then samples $k_1 \leftarrow_{\$} \mathbb{Z}_p$ and sets $R_1 := G^{-k_1}$. It sets $R := R_0 \cdot R_1$ and computes $r := \mathrm{To}\mathbb{Z}_p(R)$. It outputs $s_1 := -x \cdot r + k_1$ and R_1 to the user.

$\mathsf{User}_2(\mathsf{vk}, m, H, k_0, s_1, R_1)$**:** On input of the internal user state $(\mathsf{vk}, m, H^m, k_0)$ and the signer's message (s_1, R_1), the user computes $R := R_0 \cdot R_1$ and $s := s_1 + k_0$. The user checks that $H^m = R \cdot Y^r \cdot G^s$ where $r := \mathrm{To}\mathbb{Z}_p(R)$. As the signature on m it outputs a zero-knowledge proof of knowledge $\pi := \mathrm{PoK}\{(R, s) \colon H^m = R \cdot Y^r \cdot G^s \wedge r := \mathrm{To}\mathbb{Z}_p(R)\}$ using Π_{NR}.

$\mathsf{Verify}(\mathsf{vk}, m, \pi)$ The verification runs $\Pi_{\mathsf{NR}}.\mathsf{Verify}((m, \mathsf{vk}), \pi)$ to check that π is a valid proof of knowledge of (R, s) as above.

3.2 Scheme_2

Setup Samples group parameters $\mathsf{pp}_{\mathbb{G}}$ containing generator elements, the group order p, and a description of the conversion function $\mathrm{To}\mathbb{Z}_p$.

$\mathsf{KeyGen}(\mathsf{pp}_{\mathbb{G}})$**:** $x \leftarrow_{\$} \mathbb{Z}_p, Y := G^x$. Outputs $\mathsf{sk} = x, \mathsf{vk} = (G, H, Y)$

$\mathsf{User}_1(\mathsf{vk}, m)$**:** On input $\mathsf{vk} = (G, H, Y)$, and a message $m \in \mathbb{Z}_p$, the user samples $k_0 \leftarrow_{\$} \mathbb{Z}_p$. It then computes $R_0 := \mathsf{H}_m(m) \cdot G^{-k_0}$. Outputs a user message R_0 to the signer, and saves an internal state $(\mathsf{vk}, m, \mathsf{H}_m(m), k_0)$. The user computes a proof of knowledge π_{iss} of (m, k_0) s.t. $R_0 = \mathsf{H}_m(m) G^{-k_0}$ using Π_{iss}

$\mathsf{Sign}(\mathsf{sk}, R_0)$**:** On input of the secret key sk and a user message R_0 and a proof π_{iss}, the signer verifies π_{iss} and aborts if verification fails, then samples $k_1 \leftarrow_{\$} \mathbb{Z}_p$ and sets $R_1 := G^{-k_1}$. It sets $R := R_0 \cdot R_1$ and computes $r := \mathrm{To}\mathbb{Z}_p(R)$. It outputs $s_1 := -x \cdot r + k_1$ and R_1 to the user.

$\mathsf{User}_2(\mathsf{vk}, m, \mathsf{H}_m(m), k_0, s_1, R_1)$**:** On input of the internal user state $(\mathsf{vk}, m, \mathsf{H}_m(m), k_0)$ and the signer's message (s_1, R_1), the user computes $R := R_0 \cdot R_1$ and $s := s_1 + k_0$. The user checks that $\mathsf{H}_m(m) = R \cdot Y^r \cdot G^s$ where $r := \mathrm{To}\mathbb{Z}_p(R)$. As the signature on m it outputs a zero-knowledge proof of knowledge $\pi := \mathrm{PoK}\{(R, s) \colon \mathsf{H}_m(m) = R \cdot Y^r \cdot G^s \wedge r := \mathrm{To}\mathbb{Z}_p(R)\}$ using Π_{NR}.

$\mathsf{Verify}(\mathsf{vk}, m, \pi)$ The verification runs $\Pi_{\mathsf{NR}}.\mathsf{Verify}((m, \mathsf{vk}), \pi)$ to check that π is a valid proof of knowledge of (R, s) as above.

3.3 Scheme_3

Setup Samples group parameters $\mathsf{pp}_{\mathbb{G}}$ containing generator elements, the group order p, and a description of the conversion function $\mathrm{To}\mathbb{Z}_p$.

$\mathsf{KeyGen}(\mathsf{pp}_{\mathbb{G}})$**:** $x \leftarrow_{\$} \mathbb{Z}_p, Y := G^x, \vec{H} \leftarrow_{\$} \mathbb{G}^n$. Outputs $\mathsf{sk} = x, \mathsf{vk} = (G, \vec{H}, Y)$

$\mathsf{User}_1(\mathsf{vk}, \vec{m})$**:** On input $\mathsf{vk} = (G, \vec{H}, Y)$, and a vector of attributes $\vec{m} \in \mathbb{Z}_p^n$, the user samples $k_0 \leftarrow_{\$} \mathbb{Z}_p$. It then computes $R_0 := \vec{H}^{\vec{m}} \cdot G^{-k_0}$. Outputs a user message R_0 to the signer, and saves an internal state (vk, m, H, k_0). The user computes a proof of knowledge π_{iss} of (m, k_0) s.t. $R_0 = H^m G^{-k_0}$ using Π_{iss}

$\mathsf{Sign}(\mathsf{sk}, R_0)$**:** On input of the secret key sk and a user message R_0 and a proof π_{iss}, the signer verifies π_{iss} and aborts if verification fails, then samples $k_1 \leftarrow_{\$} \mathbb{Z}_p$ and sets $R_1 := G^{-k_1}$. It sets $R := R_0 \cdot R_1$ and computes $r := \mathrm{To}\mathbb{Z}_p(R)$. It outputs $s_1 := -x \cdot r + k_1$ and R_1 to the user.

$\mathsf{User}_2(\mathsf{vk}, m, H, k_0, s_1, R_1)$**:** On input of the internal user state $(\mathsf{vk}, m, H^m, k_0)$ and the signer's message (s_1, R_1), the user computes $R := R_0 \cdot R_1$ and $s := s_1 + k_0$. The user checks that $H^m = R \cdot Y^r \cdot G^s$ where $r := \mathrm{To}\mathbb{Z}_p(R)$. As the signature on m it outputs a zero-knowledge proof of knowledge $\pi := \mathrm{PoK}\{(R, s) \colon H^m = R \cdot Y^r \cdot G^s \wedge r := \mathrm{To}\mathbb{Z}_p(R)\}$ using Π_{NR}.

$\mathsf{Verify}(\mathsf{vk}, m, \pi)$ The verification runs $\Pi_{\mathsf{NR}}.\mathsf{Verify}((m, \mathsf{vk}), \pi)$ to check that π is a valid proof of knowledge of (R, s) as above.

3.4 Blindness

We prove blindness of the schemes presented in Sections 3.1 to 3.3.

Theorem 1. *If Π_{NR} and Π_{iss} are (computationally / statistically / perfectly) zero-knowledge, then Scheme_1, Scheme_2, Scheme_3 are (computationally / statistically / perfectly) blind.*

We stress that according to Lemma 1 the variant of Π_{iss} used in Scheme_1 and Scheme_3 is statistically zero-knowledge, whereas the variant of Π_{iss} used in Scheme_2 and Π_{NR} take over the zero knowledge notion fulfilled by SNARK.

Proof. We make a series of game hops.

Game $\mathbf{G}_1$. This is the original blindness game.

Game $\mathbf{G}_2$. In this game hop, we switch to simulating the proofs of Π_{iss} that are generated in the oracle calls to $\mathsf{User}_{1,0}$ and $\mathsf{User}_{1,1}$. This step can be bounded by the zero-knowledge property of Π_{iss}. As the challenger simulates two proofs, the game hop can be bounded by $2 \cdot \frac{Q_{\mathsf{H}_{\mathsf{iss}}}}{p}$.

Game $\mathbf{G}_3$. In this game, we switch to simulating the proofs of Π_{NR} that are generated in the oracle calls to $\mathsf{User}_{2,0}$ and $\mathsf{User}_{2,1}$. This proof step can be bounded by the zero-knowledge advantage of Π_{NR} times 2.

Game $\mathbf{G}_4$. We now are able to switch to a commitment of 0 for R_0 in both sessions. This game is identically distributed to $\mathbf{G}_3$ due to the perfect hiding property of the Pedersen commitment. As all proofs are simulated, no witness is needed to generate them.

We are now in a game where the adversary $\mathcal{S}$ has advantage 0 as both the messages msgU_0 and msgU_1 are perfectly independent of b, as well as the generated signatures are perfectly independent of b.

□

3.5 One-More Unforgeability

We prove one-more unforgeability of the schemes. The proof works the same for all three schemes plugging in their respective underlying building blocks.

Theorem 2. *If Σ_{NR} resp Σ_{mNR} is* **euf-cma** *secure, and the respective variants of Π_{NR} and Π_{iss} are witness-extended emulatable in the AGM over $\mathbb{G}'$, then it holds that Scheme_1 resp. Scheme_2, resp Scheme_3 is ℓ-***omuf***-secure for $\ell = \mathsf{poly}(\lambda)$.*

Proof. We prove this as a series of games.

Game $\mathbf{G}_1$. This game is the ℓ-**omuf** game.

Game $\mathbf{G}_2$. In the first game, we switch to extracting the witnesses from the proof Π_{iss}. The game aborts if extraction fails at any point. This game hop can be bounded by the advantage for the witness-extended emulation game. A reduction simulates either $\mathbf{G}_2$ or $\mathbf{G}_1$ to the adversary by replacing any verification of proofs of Π_{iss} during signing queries by a query to the oracle $\mathcal{O}$. If the reduction is playing in game Real, it simulates $\mathbf{G}_1$, whereas if the reduction is playing $\mathsf{Emulated}$, it simulates $\mathbf{G}_2$. It thus holds that $\mathrm{adv}^{\mathbf{G}_2}_{\mathcal{A}}(\lambda) \leq \mathrm{adv}^{\mathbf{G}_1}_{\mathcal{A}}(\lambda) + \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{iss}},\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_3$. This game generates signatures by extracting the message m along with the randomness k_0 from the adversary's proof Π_{iss}. It then signs m using the signing algorithm of Σ_{NR}. The signature is s, R. The game then "re-blinds" the signature to $s_1 = s - k_0$ and $R_1 = R \cdot G^{-k_0}$ and outputs s_1, R_1 to the adversary. The signatures are identically distributed to $\mathbf{G}_2$ and as $\mathbf{G}_2$ aborts when extraction fails, $\mathbf{G}_3$ is identically distributed to $\mathbf{G}_2$.

Game $\mathbf{G}_4$. In this game, we switch to extracting the witnesses from the proof Π_{NR}. The game aborts if extraction fails at any point. This game hop can be bounded by the advantage for the witness-extended emulation game. A reduction simulates either $\mathbf{G}_4$ or $\mathbf{G}_3$ to the adversary by replacing any verification of proofs of Π_{NR} during signature verification by a query to the oracle $\mathcal{O}$. If the reduction is playing in game Real, it simulates $\mathbf{G}_3$, whereas if the reduction is playing $\mathsf{Emulated}$, it simulates $\mathbf{G}_4$. It thus holds that $\mathrm{adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda) \leq \mathrm{adv}^{\mathbf{G}_3}_{\mathcal{A}}(\lambda) + \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{NR}},\mathcal{A}}(\lambda)$.

Game $\mathbf{G}_5$. This game hop is relevant only for Scheme_3. This game aborts if in the list of forgeries, the adversary submits two message signature pairs $(\vec{m}_1, \sigma_1)$ and $(\vec{m}_2, \sigma_2)$ such that $\vec{m}_1 \neq \vec{m}_2$ but $\vec{H}^{\vec{m}_1} = \vec{H}^{\vec{m}_2}$. This game hop can be bounded by the hardness of the discrete logarithm problem in $\mathbb{G}$. A reduction embeds a discrete logarithm challenge in one of the H_i's

and tries to solve for its discrete logarithm in case the abort condition occurs. This succeeds with probability $\frac{1}{n}$ over the choice of the embedding index and thus $\mathrm{adv}^{\mathbf{G}_5}_{\mathcal{A}}(\lambda) \leq \mathrm{adv}^{\mathbf{G}_4}_{\mathcal{A}}(\lambda) + n \cdot \mathrm{adv}^{\mathbf{DL}}_{\mathbb{G},\mathcal{R}'}(\lambda)$.

Simulating $\mathbf{G}_5$ We now provide a reduction that breaks the **euf-cma** security of Σ_{NR}. The reduction works as follows:

Setup. The reduction receives a verification key $\mathsf{vk} = (G, Y)$ resp $\mathsf{vk} = (G, H, Y)$ from the **euf-cma** game. In the case of Scheme_1 and Scheme_2 it sets $\mathsf{vk}' = \mathsf{vk}$. In the case of Scheme_3 it selects exponents $t_1, \ldots, t_n \leftarrow_\$ \mathbb{Z}_p$ and sets $\mathsf{vk}' = (G, (H_i = H^{t_i})_{i\in[n]}, Y)$ It then outputs vk to the adversary.

Online Phase. The reduction simulates the game to $\mathcal{A}$ as follows:

Random Oracles: It simulates all other random oracles by lazy sampling.

Oracle Sign It simulates the signing oracle by first extracting the message m resp. $\vec{m}$, and user randomness k_0 from the Π_{iss} proofs. In the case of Scheme_3 it computes It then queries m to the signing oracle of the **euf-cma** game and receives a signature s, R which it "re-blinds" it with k_0 as described in $\mathbf{G}_3$. It outputs the signer message s_1, R_1 to the adversary.

Output determination. When the adversary outputs its message, signature pairs, the reduction identifies the message-signature pair m^*, π^* where it has never queried m^* to the signing oracle. In the case of Scheme_3, it instead identifies the vector $\vec{m}$ with this property and sets $m^* = \sum_{i=1}^{n} m_i t_i$. By the abort in $\mathbf{G}_5$, this message was never queried. It extracts the signature σ^* from π^* and returns m^*, σ^* to the challenger.

We therefore obtain that $\mathrm{adv}^{\mathbf{G}_5}_{\mathcal{A}}(\lambda) \leq \mathrm{adv}^{\mathbf{euf\text{-}cma}}_{\Sigma_{\mathrm{NR}},\mathcal{R}}(\lambda)$.
Putting everything together we get

$$\mathrm{adv}^{\ell\text{-}\mathbf{omuf}}_{\mathcal{A}} \leq \mathrm{adv}^{\mathbf{euf\text{-}cma}}_{\Sigma_{\mathrm{NR}},\mathcal{R}}(\lambda) + \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{NR}},\mathcal{A}}(\lambda) + \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{iss}},\mathcal{A}}(\lambda) + n \cdot \mathrm{adv}^{\mathbf{DL}}_{\mathbb{G},\mathcal{R}'}(\lambda).$$

The theorem statement follows with the fact that **euf-cma**-security of Σ_{mNR} implies hardness of **DL** in $\mathbb{G}$. □

4 Zero-knowledge proofs for issuance and signatures

In this section we describe the proofs Π_{iss} used for issuance and Π_{NR} used for signatures.

4.1 Proving knowledge of a message and opening: Π_{iss}

In Scheme_1 and Scheme_3, the first user message is of the form $R = G^{-k_0} \cdot H^m$ and $R = G^{-k_0} \cdot H_1^{m_1} \cdot \ldots \cdot H_n^{m_n}$, along with a proof π_{iss} of knowledge of $(-k_0, m)$ or $(-k_0, m_1, \ldots, m_n)$, respectively.

In the following, we describe how π_{iss} is generated where we use a more general proof of partial opening, i.e., the proving algorithm may take as input also a set of indices I for which it opens the commitment, and proves knowledge of the values at the rest of the coordinates. This is useful in two ways: on the one hand, it allows the proof system to be re-used in Π_{NR} where we need to prove knowledge of a representation w.r.t. more basis elements than in Π_{iss}, thus for Π_{iss} we can simply set the corresponding coordinates to zero and open them with I. On the other hand, a blind signature scheme for vectors of messages is frequently the basis of an anonymous credential

scheme. When requesting a signature, a user may be asked to reveal some of their attributes, and blindness would is guaranteed among users with the same set of revealed attributes (and in the anonymous credential setting, unlinkability depends on which attributes are revealed when the credential is presented). We describe Π_{iss} for Scheme_3, noting that Scheme_1 can be viewed as the special case where $n = 1$.

Setup In addition to $\mathsf{pp}_{\mathbb{G}}$ (the parameters of the signature scheme), in a second group $\mathbb{G}'$ of the same order p, sample generators $G', Y', H'_1, \dots H'_n$.

Prove To prove knowledge of a private opening $(-k_0, \vec{m})$ with public indices I, sample random $t \leftarrow \mathbb{Z}_p$ and compute $R' = G'^{-k_0} Y'^{t} \prod_{i=1}^{n} H_i'^{m_i}$. This serves as a commitment to $(-k_0, \vec{m})$ in $\mathbb{G}'$. The prover now computes a Σ-proof that $(-k, \vec{m})$ are the same in R and R' (which is straightforward because R and R' are from groups of the *same* order). Sample $r_t, r_0, r_i, i \in [n] \setminus I \leftarrow_{\$} \mathbb{Z}_p$. Compute $C = G^{r_0} \prod_{i \in [n] \setminus I} H_i^{r_i}$ and $C' = G'^{r_0} Y'^{r_t} \prod_{i \in [n] \setminus I} H_i'^{r_i}$. Query $c = \mathsf{H}_{\mathsf{iss}}(R, R', C, C', I, \{m_i : i \in I\})$. Compute $s_0 = r_0 - c \cdot k_0$, $s_t = r_t + c \cdot t$ and $s_i = r_i + c \cdot m_i$ for all $i \in [n] \setminus I$. Output the proof $\pi_{\mathsf{iss}} = (R', c, s_t, s_0, \{s_i : i \in [n] \setminus I\}, \{m_i : i \in I\})$.

Verification On input (R, π_{iss}), accept the proof as valid if

$$c = \mathsf{H}_{\mathsf{iss}}\left(R, R', (R/\prod_{i \in I} H_i^{m_i})^{-c} G^{s_0} \prod_{i \in [n] \setminus I} H_i^{s_i}, \right.$$
$$\left. (R'/\prod_{i \in I} H_i'^{m_i})^{-c} G'^{s_0} Y'^{s_t} \prod_{i \in [n] \setminus I} H_i'^{s_i}, I, \{m_i : i \in I\}\right). \quad (1)$$

Completeness We show *completeness*:

$$\begin{aligned}
c =& \mathsf{H}_{\mathsf{iss}}(R, R', G^{r_0} \prod_{i \in [n] \setminus I} H_i^{r_i}, G'^{r_0} Y'^{r_t} \cdot \prod_{i \in [n] \setminus I} H_i'^{r_i}, I, \{m_i : i \in I\}) \\
=& \mathsf{H}_{\mathsf{iss}}(R, R', G^{s_0 + c \cdot k_0} \prod_{i \in [n] \setminus I} H_i^{s_i - c \cdot m_i}, \\
& G'^{s_0 + c \cdot k_0} Y'^{s_t - c \cdot t} \cdot \prod_{i \in [n] \setminus I} H_i'^{s_i - c \cdot m_i}, I, \{m_i : i \in I\}) \\
=& \mathsf{H}_{\mathsf{iss}}(R, R', G^{s_0} (R/\prod_{i \in [n]} H_i^{m_i})^{-c} \prod \prod_{i \in [n] \setminus I} H_i^{s_i}, \\
& G'^{s_0} Y'^{s_t} (R'/\prod_{i \in I} H_i'^{m_i}) \prod_{i \in [n] \setminus I} H_i'^{s_i}, I, \{m_i : i \in I\})
\end{aligned}$$

Lemma 1. *Π_{iss} is statistically zero-knowledge in the Random Oracle Model with advantage $\frac{Q_{\mathsf{H}_{\mathsf{iss}}}}{p}$ where $Q_{\mathsf{H}_{\mathsf{iss}}}$ is the number of random oracle queries the adversary makes to $\mathsf{H}_{\mathsf{iss}}$.*

Proof. The simulator takes as input a commitment R along with parameters $G, H_1, \dots, H_n$, $G', Y', H'_1, \dots H'_n$, an index set I and a partial opening $\{m_i, i \in I\}$.

It picks $k \leftarrow_{\$} \mathbb{Z}_p$ and sets $R' = G'^{k}$. It picks values $c, s_t s_0, s_i; i \in [n] \setminus I$ and sets $C = G^{s_0} \prod_{i \in [n] \setminus I} H_i^{s_i} (R/\prod_{i \in I} H_i^{m_i})^{-c}$ and $C' = G'^{s_0} \prod_{i \in [n] \setminus I} H_i'^{s_i} (R'/\prod_{i \in I} H_i^{m_i})^{-c}$.

$$\left(\begin{array}{lllllll}
\mathrm{dlog}_G C = & -c\cdot(\mathrm{dlog}_G R - \sum_{i\in I} m_i\cdot \mathrm{dlog}_G H_i) & & +s_0 & +(\mathbf{1}_{1\notin I}\cdot s_1)\cdot \mathrm{dlog}_G H_1 & +\ldots & +(\mathbf{1}_{n\notin I}\cdot s_1)\cdot \mathrm{dlog}_G H_n \\
a_y = & -c\cdot z_y & +s_t & & & & \\
a_1 = & -c\cdot z_0 & & +s_0 & & & \\
a_2 = & -c(z_1 - \mathbf{1}_{1\in I}\cdot m_1) & & & +\mathbf{1}_{1\notin I}\cdot s_1 & & \\
 & \vdots & & & & & \\
a_n = & -c\cdot(z_n - \mathbf{1}_{n\in I} m_n) & & & & & +\mathbf{1}_{n\notin I} s_n
\end{array}\right)$$

Figure 1: Linear equation system over C, R and the algebraic representations of C', R' used in the proof of Lemma 2. $\mathbf{1}_{st}$ is the function that is 1 when st holds and 0 otherwise.

It programs the random oracle $\mathsf{H}_{\mathsf{iss}}$ to output $c = \mathsf{H}_{\mathsf{iss}}(R, R', C, C', I, \{m_i : i \in I\})$ and outputs the proof $R', c, s_t, s_0, s_i; i \in [n] \setminus I$. This programming fails if the adversary has previously made a query to $\mathsf{H}_{\mathsf{iss}}$ that coincides with the simulator's query which happens with probability at most $\frac{Q_{\mathsf{H}_{\mathsf{iss}}}}{p}$ per simulated proof.

As it holds that for any R, R' and G, H_i, G', H'_i and any $k_0, m_1, \ldots, m_n$ there exists t, s_t such that $R = G^{-k_0}\prod H_i^{m_i}$ and $R' = G'^{-k_0}Y'^{t}\prod H_i'^{m_i}$, the output of the simulator is perfectly indistinguishable from that of a real prover if the random oracle programming succeeds.

□

Witness-Extended Emulation We show *witness-extended emulation in the AGM over* $\mathbb{G}'$:

Lemma 2. *If the discrete logarithm problem is hard in* $\mathbb{G}'$ *the proof system* Π_{iss} *is witness-extended emulatable in the AGM over* $\mathbb{G}'$ *and with* $\mathsf{H}_{\mathsf{iss}}$ *modelled as a random oracle.*

Proof. We provide a series of games to switch from the real game $\mathsf{Real}_{\mathcal{A}}$ to the emulated game $\mathsf{Emulated}_{\mathcal{A}}$ for which we provide the emulator $\mathcal{E}$. We stress that we consider adversaries that are algebraic over $\mathbb{G}'$ but standard (group) model adversaries over $\mathbb{G}$.

Game $\mathbf{G}_1$. This is the game $\mathsf{Real}_{\mathcal{A}}$ from Definition 13.

Game $\mathbf{G}_2$. In $\mathbf{G}_2$, we introduce an abort condition where the game aborts whenever the adversary outputs a statement-proof pair $R, I, \{m_i : i \in I\}, = (R', c, s_0, \{s_i : i \in [n]\setminus I\}$ to the oracle $\mathcal{O}$ such that $\mathsf{Verify}(R, \pi_{\mathsf{iss}}) = 1$ without having queried $c' = \mathsf{H}_{\mathsf{iss}}(R, R', (R/\prod_{i\in I} H_i^{m_i})^{-c}G^{s_0}\prod_{i=1}^n H_i^{s_i}, (R'/\prod_{i\in I} H_i^{m_i})^{-c}G'^{s_0}Y'^{s_t}\prod_{i=1}^n H_i'^{s_i}, I, \{m_i : i \in I\})$. It is easy to see that the probability of the proof verifying without a previous hash query is at most $\frac{1}{p}$ per proof as it happens with probability at most $\frac{1}{p}$ that $c' = c$ and thus, union-bounding over all oracle queries yields $\Pr[\mathbf{G}_2^{\mathcal{A}} = 1] - \Pr[\mathbf{G}_1^{\mathcal{A}} = 1] \leq \frac{Q_{\mathcal{O}}}{p}$.

Game $\mathbf{G}_3$. To define our next abort event, we consider the algebraic representation $z_0, z_1, \ldots, z_n$ submitted by the adversary such that $R' = G'^{z_0}Y'^{z_y}\prod_{i=1}^n H_i'^{z_i}$. Our future extractor will output $k_0 = -z_0, m_i = z_i$ for all $i \in [n] \setminus I$.

We analyze the probability that $R = G^{-k_0}\cdot\prod_{i=1}^n H_i^{m_i}$, i.e. that the extractor extracts the correct value.

Assume for contradiction that $R \neq G^{-k_0}\cdot\prod_{i=1}^n H_i^{m_i}$.

We consider the following linear equation system in the variables $c, s_0, \ldots, s_n$ where $a_0, \ldots, a_n$ is the algebraic representation of C' (i.e. $C' = G^{a_0}Y'^{a_y}\prod_{i=1}^n H^{a_i}$), and $\tilde{m_i}$ is m_i in the case of $i \in I$, and z_i in the case of $i \in [n] \setminus I$. The system is depicted in Figure 1 It is easy to see that for the case where $z_i \neq m_i \wedge i \in I$, there is only one possible choice of c as $a_i = -c\cdot(z_i - m_i)$ is a linear equation that fully determines c.

It remains to show that the same holds for the case that for all $i \in I$, $z_i = m_i$. This linear equation system is linearly independent in the case that $\text{dlog}_G R \neq z_0 + \sum_{i\in[n]\setminus I} z_i \text{dlog}_G H_i + \sum_{i\in I} m_i \cdot \text{dlog}_G H_i$ and thus has at most one solution $c^*, s_0^*, s_1^*, \ldots, s_n^*$. The probability that the chosen $c = \mathsf{H}_{\mathsf{iss}}(R, R', G^{s_0}\prod_{i=1}^n H_i^{s_i} R^{-c}, G'^{s_0}\prod_{i=1}^n H_i'^{s_i} R'^{-c}, I, \{m_i : i \in I\}) = c^*$ is $\frac{1}{p}$ per random oracle query.

In $\mathbf{G}_3$, we introduce an abort in the case that $c = \mathsf{H}_{\mathsf{iss}}(R, R', G^{s_0}\prod_{i=1}^n H_i^{s_i} R^{-c}, G'^{s_0}\prod_{i=1}^n H_i'^{s_i} R'^{-c}, I, \{m_i : i \in I\}) = c^*$ while $R \neq G^{-k_0} \cdot \prod_{i=1}^{m_i} H_i^{m_i}$, which by union bound over the queries to $\mathsf{H}_{\mathsf{iss}}$ yields the following bound:$\Pr[\mathbf{G}_3^{\mathcal{A}} = 1] - \Pr[\mathbf{G}_2^{\mathcal{A}} = 1] \leq \frac{Q_{\mathsf{H}_{\mathsf{iss}}}}{p}$.

Game $\mathbf{G}_4$. In $\mathbf{G}_4$, we abort if the the adversary outputs a valid proof despite $R \neq G^{-k_0} \cdot \prod_{i=1}^n H_i^{\tilde{m}_i}$ and $c \neq c^*$ where $\tilde{m}_i$ is m_i for $i \in I$ and the extracted value for m_i for $i \notin I$. To bound this game hop, we show that if the discrete logarithm problem is hard in $\mathbb{G}'$, it is infeasible for the adversary to provide a valid proof in the case that $c \neq c^*$.

The reduction picks whether to embed its discrete logarithm challenge in H_i' for some i or in Y. It samples exponents $u_j \leftarrow_\$ \mathbb{Z}_p$ for $j \in [n], j \neq i$. When the adversary outputs a proof $\pi_{\mathsf{iss}} = (R', c, s_t, s_0, \{s_i : i \in [n] \setminus I\}, \{m_i : i \in I\})$. , the reduction computes $\text{dlog}_{G'} H_i' = \frac{s_0 - cz_0 + (s_t - cz_y)\cdot \text{dlog}_{G'} Y' \sum_{j\neq i}(s_j - cz_j)\cdot \text{dlog}_{G'} H_j'}{a_i - s_i + cz_i}$ or $\text{dlog}_{G'} Y' = \frac{s_0 - cz_0 + \sum_{j\in[n]}(s_j - cz_j)}{a_y - s_t + cz_y}$. As $c \neq c^*$, the denominator is non-zero for some i and thus with probability $\frac{1}{n}$ the reduction solves the discrete logarithm problem.

It therefore holds that $\Pr[\mathbf{G}_4^{\mathcal{A}} = 1] - \Pr[\mathbf{G}_3^{\mathcal{A}} = 1] \leq n \cdot \text{adv}^{\mathbf{DL}}_{\mathbb{G}',\mathcal{R}}(\lambda)$.

Game $\mathbf{G}_5$. We switch to using the following emulator $\mathcal{E}$ in the oracle $\mathcal{O}$, i.e. $\mathbf{G}_5 = \mathsf{Emulated}_{\mathcal{A}}$: We provide an emulator $\mathcal{E}$ in the random oracle model such that for any algebraic adversary $\mathcal{A}$ over $\mathbb{G}'$ that outputs R, π_{iss} such that whenever $\mathcal{A}$ outputs a valid proof π_{iss} and $\mathbf{G}_4$ doesn't abort, the emulator $\mathcal{E}$ outputs $(-k_0, m_1, \ldots, m_n)$ such that $R = G^{-k_0}\prod_{i=1}^n H_i^{m_i}$ with probability $\varepsilon - \text{adv}^{\mathbf{DL}}_{\mathcal{R},\mathbb{G}'}(\lambda) - \frac{Q_{\mathsf{H}_{\mathsf{iss}}}}{p}$ where $\mathcal{R}$ is a reduction against **DL** in the group $\mathbb{G}'$ and $Q_{\mathsf{H}_{\mathsf{iss}}}$ is the number of queries that $\mathcal{A}$ makes to $\mathsf{H}_{\mathsf{iss}}$.

The emulator receives the group element inputs from $\mathbb{G}'$ to the RO queries that the adversary makes to $\mathsf{H}_{\mathsf{iss}}$ along with a list of the representations of each element. The emulator $\mathcal{E}$ aborts in the abort conditions described above. It is easy to see that the emulator simulates the oracle $\mathcal{O}$ of $\mathbf{G}_4$ perfectly and thus $\Pr[\mathbf{G}_5^{\mathcal{A}} = 1] = \Pr[\mathbf{G}_4^{\mathcal{A}} = 1]$. □

4.2 Proving knowledge of a message in Scheme_2

In Scheme_2 the user's messsage during issuance is computed $R_0 = G^{-k_0}\mathsf{H}(m)$, where H is traditional hash function whose outputs are in $\mathbb{G}$, as defined in the standard Nyberg-Rueppel signature scheme. We assume $\mathbb{G}$ is an elliptic curve group, set H to be SHA-256, and also assume the user can vary m (say with a counter, or by choosing it at random) until $\mathsf{H}(m)$ when encoded as a field element is the x-coordinate of a point in $\mathbb{G}$. The challenge of realizing Π_{iss} for Scheme_2 is efficiently proving knowledge of a SHA-256 preimage and the random coins k_0 used in the commitment. We use Spartan for this case.

We want to prove knowledge of (m, k_0, t) such that R, R' fulfill:

$$R = G^{-k_0} \cdot M \quad \wedge \quad R' = G'^{-k_0} \cdot H'^{m} \cdot Y^{t} \quad \wedge \quad M = \mathsf{H}_m(m)$$

We first describe the scheme $\Pi_{\mathsf{NR}}{}^{(2)}$ in a bit more detail:

Setup In addition to $\mathsf{pp}_{\mathbb{G}}$ (from the signature scheme), in a second group $\mathbb{G}'$ of the same order p, sample generators G', H', Y'. Run $\mathsf{SNARK.Setup}$

Prove Sample $t \leftarrow_\$ \mathbb{Z}_p$ and set $R' = G'^{-k_0} H'^m Y'^t$. Further sample $r_0, r_t, r_m \leftarrow_\$ \mathbb{Z}_p$ and set $C' = G'^{r_0} Y'^{r_t} H'^{r_m}$. Compute $h_r = \mathsf{H}_m(r_0, r_t, r_m)$ Run $c = \mathsf{H}_{\mathsf{iss}}(R, R', C, h_r)$ and compute $s_0 = r_0 + c \cdot k_0$, $s_t = r_t - c \cdot t$ and $s_m = r_m - c \cdot m$. Output the proof (c, s_0, s_t, s_m) along with π_{SNARK} proving that $(C, C'h_t, h_k) \in L_{\mathsf{SNARK}}$.

Verify Verification checks the Σ-proof in the obvious way and runs the verification algorithm of SNARK on π_{SNARK}. It accepts if both parts accept.

Lemma 3. *If the SNARK is knowledge-sound and the discrete logarithm problem is hard in $\mathbb{G}'$, then Π_{iss} is witness extended emulatable in the AGM over $\mathbb{G}'$.*

Proof. The extractor for WEE will proceed as follows: When provided with the proof, the extractor will check that the proof verifies. If so, it takes the algebraic representation of the proof part R'. It tests whether $G^{k_0} \cdot \mathsf{H}_m(m) = R$ and if so outputs the representation as the values k_0, m, t.

We will now prove that WEE holds for this extractor in a series of games.

Game $\mathbf{G}_1$. This is the real game where the adversary interacts with a verification oracle.

Game $\mathbf{G}_2$. In this game, we switch to extracting a witness as follows: Take the representation of the commitment C' and consider m to be the exponent of H' and $-k_0$ to be the exponent of G'. If for the extracted m and k_0 it doesn't hold that $R = H(m) \cdot G^{-k_0}$, abort this game. We bound the distance between the games:

Claim 4.1. $\Pr[\mathbf{G}_2 = 1] \geq \Pr[\mathbf{G}_1 = 1] - \frac{2\mathrm{adv}^{\mathbf{DL}}_{\mathbb{G}',\mathcal{R}}(\lambda)}{(1-\kappa(\lambda))}$.

Proof. We provide a reduction that solves a discrete logarithm challenge in $\mathbb{G}'$ if an adversary triggers the abort condition. The reduction embeds its discrete logarithm challenge in either H' or V' and samples the other group element with a known discrete logarithm. It generates $G, H \in \mathbb{G}$ as in the key generation algorithm of Scheme_2.

When the adversary outputs a proof π_{iss} that triggers the abort condition of $\mathbf{G}_2$, the reduction runs the extractor $\mathcal{E}_{\mathsf{SNARK}}$ on the adversary. When the extractor $\mathcal{E}_{\mathsf{SNARK}}$ outputs a witness (m', k'_0, t'), this witness is guaranteed to fulfill $((R, R'), (m', k'_0, t')) \in \mathcal{R}_{\mathsf{SNARK}}$ with probability $1 - \kappa(\lambda)$. Thus, the witness is different from the algebraic representation submitted earlier by the adversary. The reduction uses the two different representations to solve for the discrete logarithm in $\mathbb{G}'$. This succeeds with probability at least $\frac{1}{2}$ over the choice of where the challenge was embedded. Putting things together yields the claim. □

It remains to see that $\mathbf{G}_2$ is identical to the emulated game. This is the case as both $\mathbf{G}_2$ and the emulator extract the witness in the same manner and check whether it fulfills the statement of SNARK. □

Lemma 4. *If SNARK is (perfectly / statistically / computationally) zero-knowledge, then Π_{iss} for Scheme_2 described above is (perfectly / statistically / computationally) zero-knowledge.*

Proof. We describe a simulator. To generate the secondary commitment C' in $\mathbb{G}'$, it picks a random group element from $\mathbb{G}'$ The simulator then uses the Σ-protocol simulator in $\mathbb{G}'$ to generate a transcript for the proof of knowledge of discrete logarithms in $\mathbb{G}'$. It then inputs the transcript along with the public inputs into the simulator of the SNARK to obtain the proof part π_{SNARK}. It is easy to see that zero-knowledge follows from the zero-knowledge properties of the underlying properties. □

4.3 Proving knowledge of a signature: Π_{NR}

To prove knowledge of a signature, we must prove knowledge of values satisfying the verification equation. The verification equation for our NR-based blind signature schemes is mostly algebraic, and amenable to proof with well-known Σ-protocols for proving knowledge of representations of group elements. However, verification must also check a relationship between group elements and scalars (using the conversion function), and this step is expressed more naturally for circuit-based proof systems like zkSNARKS, than Σ-protocols. Previous approaches using only Σ-proofs for proving similar relations (knowledge of ECDSA signatures) ZKAttest [FLM22] and CDLS [CLR24] are less efficient than our approach, having non-interactive proofs that are at least 150 KB. We pass over these techniques due to the large proof size, and note that the prover and verifier times are likely also higher than our approach (based on the CDLS times reported in [CLR24]).

We use a mixed approach, where a Σ-proof is used for most of the algebraic part of verification, and a zkSNARK is used for the conversion function. The two proofs are bound together using the Π_{dlhash} proof technique of [OKMZ25].

Scheme_1 In Scheme_1 we have a signature (R, s) and must prove it satisfies the verification equations, $R = Y'^{-r} G^s H^m$ and $r = \mathrm{To}\mathbb{Z}_p(R)$. The values (Y', G, H, m) are public, and (R, r, s) must remain private. The relation is thus:

$$W_{\mathrm{NR}} = \left\{((Y', G, H, m), (R, r, s)) : R = Y'^{-r} G^s H^m \ \wedge \ r = \mathrm{To}\mathbb{Z}_p(R).\right\}$$

The proof strategy is as follows. Let V be a random element of $\mathbb{G}$ (a fixed parameter).

1. Sample $z \in \mathbb{Z}_p$, and randomize the algebraic part of the verification equation:

$$RV^z = Y'^{-r} G^s H^m V^z \tag{2}$$

   Let B denote this group element, the prover makes B public.

2. Use the Π_{dlhash} proof of [OKMZ25]. This proof has two parts:

   (a) A Σ-proof of the representation $(-r, s, m, z)$ of B with respect to the bases (Y', G, H, V). (RHS of Equation (2)) Note that since m is public, we prove that $B/H^m = Y'^{-r} G^s V^z$.

   (b) A SNARK proof that

      i. $R = B/V^z$ (LHS of Equation (2))

      ii. $r = \mathrm{To}\mathbb{Z}_p(R)$

      iii. (plus some additional checks specified by Π_{dlhash} that bind the Σ and SNARK proofs)

The SNARK in Step 2b is the most expensive part of the overall proof, in particular Step 2(b)i requires computing a scalar multiplication in the circuit. To minimize this cost we will instantiate the proof system so that it uses arithmetic circuits in $\mathbb{Z}_q$ where q is the prime used to implement arithmetic in $\mathbb{G}$. When $\mathbb{G}$ is an elliptic curve E defined over $\mathbb{F}_q$, we will instantiate the proof system with a curve E' over $\mathbb{F}_r$ that has $\#E' = q$. This is sometimes called a *chain* of elliptic curves, or a *half-cycle*, and is an easy condition to meet efficiently (unlike, say full cycles between pairing-friendly curves). It's usually the case that one in the chain is faster, since the prime defining the base field can be chosen to have a special form, whereas for the other curve since the group order is fixed the base field is not special. For our application there is much more

arithmetic done on the curve in the proof system than in the signature scheme, meaning we'd prefer to have the proof system use the most optimized curve and the signature scheme using the other curve. For example, a fast choice would be to have curve25519 used by the proof system, and signatures generated on the related curve T-25519 (this pair of curves was used in the opposite order in [WOS+25] to prove knowledge of EdDSA signatures on curve Ed25519).

Scheme_2 In Scheme_2 the first part of the verification equation changes to $R = Y'^{-r} G^s H(m)$. Since m is public, $H(m)$ is computed by the prover and verifier and input to proof generation and verification. Thus Step 2a is slightly modified, the prover proves knowledge of $(-r, s, v)$ such that $B/H(m) = Y'^{-r} G^s V^z$. Overall the efficiency of the proof is nearly identical.

Scheme_3 In Scheme_3, when the vector of messages $\vec{m} = (m_1, \ldots, m_n)$ is signed, the verification equation becomes

$$R = Y'^{-r} G^s \prod_{i=1}^{n} {H_i}^{m_i}$$

The randomization in Step 1 works as before, but the Σ-proof of Step 2a is modified in a straightforward way to prove knowledge of $\vec{m}$. An easy modification to the Σ-proof is to have subsets of $\vec{m}$ be revealed, hidden and committed (in fresh Pedersen commitments), to get the basic functionality of an anonymous credential system. Further, using Π_{dlhash} allows us to prove complex predicates about the hidden attributes. For example, to limit the number of times a credential may be used to k times, we can prove that the public value t is computed as $t = \mathsf{PRF}(m_0, c)$ and prove that $c \in [k]$. Since the PRF is deterministic, the prover can only compute k distinct tag values t. A verifier that logs tag values can enforce that every authentication uses a fresh tag, thereby limiting each credential to k uses. When Poseidon is used to implement the PRF, this adds only a small number of constraints to the circuit (approximately $300 + \log_2(c)$).

4.3.1 Witness Extended Emulation of Π_{NR}

Lemma 5. *If the discrete logarithm problem is hard in $\mathbb{G}'$, and the SNARK is knowledge-sound with knowledge-error $\kappa \leq \mathrm{negl}$, and the protocol Π_{iss} is witness extended-emulatable in the AGM over G' with $\mathsf{H}_{\mathsf{iss}}$ modelled as a random oracle, then the scheme Π_{NR} described above is witness extended-emulatable in the AGM over $\mathbb{G}'$ with $\mathsf{H}_{\mathsf{iss}}$ modelled as a random oracle.*

Proof. We provide an emulator: The emulator first runs the extraction procedure of Π_{iss} to obtain a witness $(-r, s, m, z)$, resp. $(-r, s, \vec{m}, z)$. If extraction fails, return false. It then verifies the points listed in Item 2b and returns true if they all hold, false otherwise.

We show that this emulator is indistinguishable from the real verification oracle. We proceed in two game hops.

Game $\mathbf{G}_1$. This is the real world game Real.

Game $\mathbf{G}_2$. In this game, we switch to extracting using the emulator from Π_{iss}. If the emulator returns false, the game returns false. This game hop can be bounded by the witness extended emulation of Π_{iss}.

Game $\mathbf{G}_3$. In this game, we switch to verifying the points from Item 2b manually. The game aborts if the extracted witness from the previous game does not fulfill the statement of the SNARK.

We show that the abort condition can be bounded by the binding property of the commitment scheme and the knowledge soundness of the SNARK.

A reduction first proceeds as follows: The reduction takes as input a discrete logarithm challenge X' and embeds it in one of the group elements of Y', V', H', resp. $Y', G', V', \vec{H}'$ and samples the other group elements like in an honest execution of the protocol.

Build a wrapper $\mathcal{A}'$ around $\mathcal{A}$ that takes the same inputs as $\mathcal{A}$. The wrapper simulates $\mathbf{G}_2$ to the adversary and runs the Π_{iss}-extractor to extract candidate witnesses. When the extracted witness does not match the SNARK statement, it outputs the SNARK proof π and the mismatched witness $(-r', s', v')$.

The reduction runs the SNARK knowledge soundness extractor $\mathcal{E}_{SNARK}$ on $\mathcal{A}'$. It outputs a witness $(-r, s, v)$, with probability $1 - \kappa(\lambda)$.

The reduction then attempts to solve for the discrete logarithm of X' using the two different witnesses. This succeeds with probability $\frac{1}{3}$ resp. $\frac{1}{n}$ over the choice of which group element X' was embedded in.

Game $\mathbf{G}_4$. In this game we switch to fully using the emulator. This game is identically distributed as $\mathbf{G}_3$.

Summing up the probabilities we get

$$\mathrm{adv}_{\Pi_{\mathsf{NR}},\mathcal{A}} \leq \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{iss}},\mathcal{A}}(\lambda) + \frac{3\mathrm{adv}^{\mathbf{DL}}_{\mathbb{G}',\mathcal{R}}(\lambda)}{(1-\kappa(\lambda))}$$

resp.

$$\mathrm{adv}_{\Pi_{\mathsf{NR}},\mathcal{A}} \leq \mathrm{adv}^{\mathbf{wee}}_{\Pi_{\mathsf{iss}},\mathcal{A}}(\lambda) + \frac{(n+1)\mathrm{adv}^{\mathbf{DL}}_{\mathbb{G}',\mathcal{R}}(\lambda)}{(1-\kappa(\lambda))}$$

□

4.3.2 Zero-Knowledge

Lemma 6. *If* SNARK *is (perfectly / statistically / computationally) zero-knowledge, then* Π_{NR} *for* Scheme_2 *described above is (perfectly / statistically / computationally) zero-knowledge.*

The proof follows the same strategy as the proof of Lemma 4.

4.4 Details of Π_{NR}

In Figure 2 we describe the proof Π_{NR} in detail. As described above, there are two main parts, following Π_{dlhash}, a Σ proof and a zkSNARK. At the begining of the Σ-proof we compute the commitment $h = H_p(r, \tilde{r})$ to r and the randomness used to hide it. The hash function H_p is Poseidon, chosen to be efficient in our SNARK (p is the base field of the signature curve and the group order of the proof system's curve, and q is the group order of the signature curve). Then the Σ-proof proceeds as expected, noting that $B = B_0 H^m$ (in Scheme_1) or $B = B_0 H(m)$ (in Scheme_2). The zkSNARK circuit then computes the checks (i)-(iv). Note that check (i) is done modulo q, and so we must emulate this arithmetic since our proof system only supports arithmetic modulo p. We use the `bellpepper-emulated` crate [Vij24] for this step. In various places we must convert values from $\mathbb{Z}_q$ to $\mathbb{Z}_p$, which is simple because we chose parameters so that $q < p$.

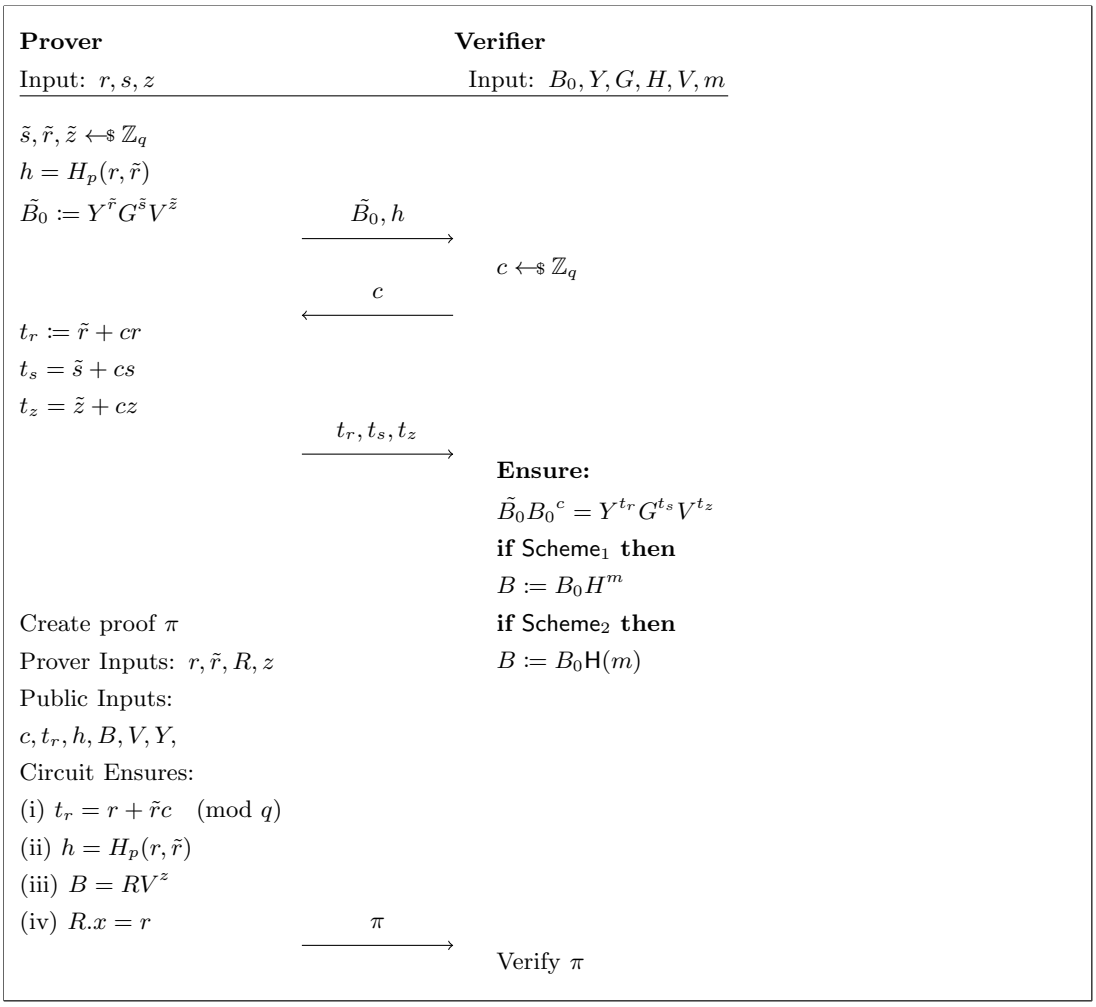

Figure 2: Detailed description of Π_{NR}. The prover also has the (public) inputs of the verifier. The hash function $H_p : \mathbb{Z}_p^2 \to \mathbb{Z}_p$ is ZK-friendly, our implementation uses Poseidon. Inputs to H_p must be converted from $\mathbb{Z}_q$ to $\mathbb{Z}_p$, which is trivial since we assume $q < p$.

5 Implementation and Performance Evaluation

We implement Scheme_1 and Scheme_2 to demonstrate the practicality of our approach, and in this section we describe our implementation, and give benchmark results.[2]

Our circuit must compute one scalar multiplication, a ZK-friendly hash (Poseidon [GKR+21]), and must emulate one field multiplication (with a different prime, sometimes called *foreign* or *nonnative* arithmetic). This circuit requires fewer constraints than the two scalar multiplications that are required to implement the verification equation directly. The two proofs are bound together using the Π_{dlhash} proof technique of [OKMZ25]. For the zkSNARK portion our implementation has two options, both relying only on discrete logarithms in prime-order groups for security, but with different performance characteristics.

The first option is based on Bulletproofs (BP) [BBB+18] and it gives the shortest proofs, at around 1.3 KB, but has slower prover times (in the range of 300-500ms). Since we express our circuit as R1CS, we implement a layer on top of the Bulletproofs inner-product argument (IPA) defined in a recent paper by Segev [Seg25] (that in turn builds on work by Bünz [Bün23]). Our implementation also outperforms previous implementations of the BP-IPA by a factor of 2-3 for large instances, by avoiding the individual scalar multiplications required to update the parameters at each recursive step, and instead pushing this update into the multi-scalar multiplications in subsequent recursive steps. This is not an asymptotic improvement, but is significant in our application. In Appendix A we provide additional details of this optimization.

Our second zkSNARK option is the Spartan proof system [Set20], in particular the Spartan-NIZK proof system, which is not asymptotically succinct, but gives better performance (both size and speed) than the succinct version (SpartanSNARK). We use a fork of the Spartan implementation that supports our R1CS programming model (bellpepper [Fra25]), supports the elliptic curves we need, and since Spartan uses the BP-IPA as a subroutine, we use our optimized IPA implementation there as well. The Spartan prover is 1.4–1.7x faster than the BP option, verifier times are the same, and proof sizes are about 10x larger. Thus for the small circuit (3873 R1CS constraints) used for proving knowledge of a signature the BP option is generally preferable. However, we also use Spartan in Scheme_2 for the proof Π_{iss} during issuance, which requires proving a SHA-256 preimage and has too many constraints for BP (28 202), but Spartan's better scaling means this proof is only slightly more costly than the Π_{NR} proof.

In order to make scalar multiplication efficient in the circuit, we choose the elliptic curve for our proof system and signature scheme jointly, as a chain of elliptic curves (also called a half-cycle). Namely, we want the proof system to efficiently prove arithmetic circuits modulo p, where the signature scheme is defined for a curve $E(\mathbb{F}_p)$. Then we can prove scalar multiplications efficiently, using techniques from existing implementations (see [WOS+25] for references and details). We choose two parameter sets, one aiming to be more conservative, and the other aiming to be more performant. In the first, signatures are computed with the NIST P-256 elliptic curve [CMR+23], and proofs on the curve T-256 [FLM22]. In this case our the assumption that the NR signature scheme is secure on the P-256 seems very mild. In the second parameter set, our choice aims to make proofs fast; we use the Pasta curves [Hop20a], a cycle of curves (named Pallas and Vesta), which have efficient endomorphisms and a high-performance, parallelized implementation of Pippenger's multi-scalar multiplication algorithm (via the `pasta-msm` crate [Pol23]). Signatures are on the Pallas curve, which is newer than the P-256 curve, but was chosen according to generally accepted (and arguably stricter) criteria for curve selection [Hop20b], and is also important for the security of the Zcash cryptocurrency. We use a fork of the `halo2curves` [Pri25] library where we add support for T-256 and modify the pasta MSM code to use `pasta-msm`. We investigated using fixed-base scalar multiplication optimizations, and concluded it is nontrivial

[2] Our source code is available at `https://github.com/zaverucha/signature-proof/`.

to apply in our setting, and may not provide a significant improvement (see Appendix B for details).

As most current processors have multiple cores, our implementation uses parallelism to accelerate the MSM computations, and this reduces the wall-clock time required for proof generation significantly. Our benchmarks use six cores, and we expect that additional cores could further reduce prover time. Section 5 gives detailed benchmarks showing the time required for each

	Operation	Time (ms)		Size (bytes)
		p256	Vesta	
Scheme_1	Issuance, user request	1.04	1.07	140
	Issuance, signer	0.93	0.77	76
	Issuance, user finalize	0.72	0.61	
	User show (BP)	503.58	342.84	1 349
	Verify show (BP)	91.47	72.61	
	User show (Spartan)	347.70	196.90	12 382
	Verify show (Spartan)	104.60	70.07	
Scheme_2	Issuance, user request	364.46	244.50	16 590
	Issuance, signer	164.28	126.31	76
	Issuance, user finalize	0.43	0.42	
	User show (BP)	507.94	341.67	1 349
	Verify show (BP)	91.75	73.07	
	User show (Spartan)	350.82	197.20	12 372
	Verify show (Spartan)	104.66	71.03	

Table 2: Benchmarks showing computation time and communication costs for Scheme_1 and Scheme_2 for the two choices of ellpitic curves (p256 and Vesta). The sizes for both curves are nearly identical, we give the sizes for p256.

sub-operation required for a user to be issued a blind signature from a signer and to then show it to a verifier. We also report the sizes of the messages communicated between the user and signer (during issuance), and the user and the verifier (when showing a signature). All benchmarks are given on an Intel Xeon W-2133 CPU @ 3.60GHz.

Scheme_1 has extremely efficient issuance, owing to the very lightweight proof Π_{iss} required in this case, whereas Scheme_2 requires a SNARK for issuance increasing the computation time and communication cost per signature. The relative speed and sizes of the two SNARK choices is discussed above, and the same speed/size tradeoff is present in the benchmarks for showing and verifying a signature. We also see the impact of the choice of curve; the Pasta curves are 1.2–1.7x faster, depending on the operation with the large gap (e.g., during proof generation) owing to the faster MSM performance of the Pasta curves and implementation. In Scheme_1 since the cost of verifying a signature is about the same with either SNARK, we would expect the BP option to be preferable for a busy server since it reduces bandwidth significantly, and increases cost on the user side (where proofs are computed relatively infrequently). Interestingly, with this same issuer-centric view, the cost of using the more standard P-256 curve is only a 1.25x slower verification.

Comparison We are not aware of any directly comparable schemes in the literature, namely round-optimal schemes that rely only on prime-order groups. Multiple well-known options exist for blind signatures with parings or RSA (e.g., [DJW23], [LKWL25]), and for those that are not round-optimal. Issuance performance of these schemes is comparable to our Scheme_1 and our scheme has slower costs to show a signature and verify the showing, as we use a SNARK for showing.

The work [FW24] is similar in that it uses a zkSNARK during issuance to create a blinded Schnorr signature and reduces security to the non-blind Schnorr signature scheme. However, the SNARK is pairing-based (Groth16 or Plonk) and the first message from user to issuer is an encryption of the message and user randomness. So while the signature scheme is based on prime-order groups (Schnorr), the security of the blind signature depends on pairing-based groups. The scheme requires over a million R1CS constraints when the signature scheme is instantiated with a standard curve and hash function (secp256k1 and SHA-256) and requires tens of seconds (Groth16, non-transparent setup) or minutes (Plonk, transparent setup). The encryption step results in a scheme with computational blindness, and for post-quantum privacy encryption must be done with PQ-secure scheme. The advantage of [FW24] is that the final signature value is a standard Schnorr or EdDSA signature, meaning we can have blind issuance of signatures that are used in existing protocols.

Acknowledgments

Julia Kastner is supported by the Dutch Research Agenda (NWA) project HAPKIDO (Project No. NWA.1215.18.002), which is financed by the Dutch Research Council (NWO). Stefano Tessaro's research was partially supported by NSF grants CNS-2026774, CNS-2154174, CNS-2426905, a gift from Microsoft, and a Stellar Development Foundation Academic Research Award.

References

[AdM04] Giuseppe Ateniese and Breno de Medeiros. A provably secure Nyberg-Rueppel signature variant with applications. Cryptology ePrint Archive, Report 2004/093, 2004.

[App21] iCloud Private Relay Overview. https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF, 12 2021. Accessed: 2026-01-13.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[BHKR26] Nicholas Brandt, Dennis Hofheinz, Michael Klooß, and Michael Reichle. Tightly-secure blind signatures in pairing-free groups. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025*, pages 337–369, Singapore, 2026. Springer Nature Singapore.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[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.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor,

PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer Berlin Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[Bün23] Benedikt Bünz. Improving the privacy, scalability, and ecological impact of blockchains (Ph.D Thesis). Technical report, Stanford University, 2023.

[CATZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Pairing-free blind signatures from CDH assumptions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 174–209, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Santa Barbara, CA, USA, August 21–25, 1990. Springer, New York, USA.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[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, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 710–742, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[CLR24] Sofía Celi, Shai Levin, and Joe Rowell. CDLS: Proving knowledge of committed discrete logarithms with soundness. In Serge Vaudenay and Christophe Petit, editors, *AFRICACRYPT 24: 15th International Conference on Cryptology in Africa*, volume 14861 of *Lecture Notes in Computer Science*, pages 69–93, Douala, Cameroon, July 10–12, 2024. Springer, Cham, Switzerland.

[CMR+23] Lily Chen, Dustin Moody, Karen Randall, Andrew Regenscheid, and Angela Robinson. Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. NIST Special Publication 800-186, 2 2023.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany.

[DJW23] Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. RFC 9474, 10 2023.

[DKT26] Marian Dietz, Julia Kastner, and Stefano Tessaro. On the impossibility of round-optimal pairing-free blind signatures in the ROM. Cryptology ePrint Archive, 2026.

[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, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[dVYA23] Henry de Valence, Cathie Yun, and Oleg Andreev. A pure-Rust implementation of Bulletproofs using Ristretto, 2023. https://github.com/dalek-cryptography/bulletproofs.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer Berlin Heidelberg, Germany.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[FLM22] Armando Faz-Hernández, Watson Ladd, and Deepak Maram. ZKAttest: Ring and group signatures for existing ECDSA keys. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021: 28th Annual International Workshop on Selected Areas in Cryptography*, volume 13203 of *Lecture Notes in Computer Science*, pages 68–83, Virtual Event, September 29 – October 1, 2022. Springer, Cham, Switzerland.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology – AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251, Gold Coast, Queensland, Australia, December 13–16, 1993. Springer Berlin Heidelberg, Germany.

[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, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[Fra25] François Garillot, Lurk Labs, and other contributors. bellpepper: SNARK Circuit library inspired by bellman/bellperson, 4 2025. https://github.com/lurk-lang/bellpepper.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.

[FW24] Georg Fuchsbauer and Mathias Wolf. Concurrently secure blind Schnorr signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 124–160, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 519–535. USENIX Association, August 11–13, 2021.

[Goo] VPN by Google One, explained. https://one.google.com/about/vpn/howitworks.

[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, 10 2021. Work in Progress.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 345–375, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[Hop20a] Daira Hopwood. The Pasta curves for Halo 2 and beyond. Electric Coin Co. Blog Post, nov 2020. https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/.

[Hop20b] Daira-Emma Hopwood. Generator and supporting evidence for security of the Pallas/Vesta pair of elliptic curves, 2020. https://github.com/zcash/pasta.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 543–571, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[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: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 468–497, Virtual Event, March 8–11, 2022. Springer, Cham, Switzerland.

[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, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 210–245, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[KR25] Michael Klooß and Michael Reichle. Blind signatures from proofs of inequality. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part VI*, volume 16005 of *Lecture Notes in Computer Science*, pages 157–189, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[KRW24] Michael Klooß, Michael Reichle, and Benedikt Wagner. Practical blind signatures in pairing-free groups. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part I*, volume 15484 of *Lecture Notes in Computer Science*, pages 363–395, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 171–189, Santa Barbara, CA, USA, August 19–23, 2001. Springer Berlin Heidelberg, Germany.

[LKWL25] Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-09, Internet Engineering Task Force, 07 2025. Work in Progress.

[Lys23] Anna Lysyanskaya. Security analysis of RSA-BSSA. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 251–280, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

[NR93] Kaisa Nyberg and Rainer A. Rueppel. A new signature scheme based on the DSA giving message recovery. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 58–61, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[Oka94] Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74, Santa Barbara, CA, USA, August 21–25, 1994. Springer Berlin Heidelberg, Germany.

[OKMZ25] Michele Orrù, George Kadianakis, Mary Maller, and Greg Zaverucha. Beyond the circuit: How to minimize foreign arithmetic in ZKP circuits. *IACR Communications in Cryptology (CiC)*, 2(1):23, 2025.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany.

[Pol23] Andy Polyakov. Pasta multi-scalar multiplication, 2023. https://github.com/supranational/pasta-msm.

[Pri25] Privacy Scaling Explorations. halo2curves: A collection of Elliptic Curves for ZkCrypto traits, 2025. https://github.com/privacy-ethereum/halo2curves.

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

[Seg25] Gil Segev. Bulletproofs for R1CS: Bridging the completeness-soundness gap and a ZK extension. Cryptology ePrint Archive, Report 2025/327, 2025.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[Tru25] Trust tokens. https://developer.chrome.com/docs/privacy-sandbox/trust-tokens/, 10 2025. Accessed: 2026-01-13.

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 782–811, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[Vij24] Saravanan Vijayakumaran. bellpepper-emulated: Nonnative arithmetic library, 4 2024. https://github.com/lurk-lab/bellpepper-gadgets/tree/main/crates/emulated.

[WJCT21] John Wilander, Frederic Jacobs, Kate Cheney, and Jiewen Tan. PCM: Click fraud prevention and attribution sent to advertiser. https://webkit.org/blog/11940/pcm-click-fraud-prevention-and-attribution-sent-to-advertiser/, 7 2021. Accessed: 2021-09-30.

[WOS+25] Anna Pui Yung Woo, Alex Ozdemir, Chad Sharp, Thomas Pornin, and Paul Grubbs. Efficient Proofs of Possession for Legacy Signatures . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 80–80, 5 2025. https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00080.

A Our IPA Implementation

In this section we provide additional details of our optimized IPA implementation, of potential application to other uses of IPAs in the literature. The prover cost of the Bulletproofs IPA is dominated by the time required to update the generators at each recursive step. This is described in the paper [BBB+18] and implemented in [dVYA23] as $n/2^i$ independent scalar multiplications (at each recursive step the input size is halved). This is natural because the generators are required for the computations of the next recursive step.

In our implementation instead of computing an updated generator $G' = G^x$ we "defer" the update to the next recursive step. We represent G' as (G, x) and then when we have to compute G'^y we first compute $y' = xy$ and then compute G'^{xy}. Further, since the generators are always used in multi-scalar multiplications, we implicitly update the generators during computation required for the proof in the recursive step in a single MSM. This has the drawback that the generators at each recursive step are always represented in terms of the original n input generators (we never collapse a term (G, x) down to G'), so the MSMs at each step has size n, rather than size $n/2^i$. However, the cost of MSMs grows much slower than the same number of single scalar multiplications and the MSM implementation parallelizes very well. We did not perform a careful cost analysis, but empirically determined that this is a good strategy on multi-core systems when compared to the approach of in [dVYA23]. We computed the benchmarks in Table 3 on our six core Intel Xeon W-2133 CPU @ 3.60GHz, with the T-256 elliptic curve.

We see that (as expected) parallelizing the loops and MSM implementation of [dVYA23] improves it significantly, especially as the instance size grows large enough to offset of the overheads of parallelization. Our new implementation strategy gives a further 1.5–3.2x speedup. Our verifier implementation is always the same, and uses the MSM-based implementation described in [BBB+18] and [dVYA23], but is parallelized (which explains why the small instance is actually a bit slower).

A more thorough study the prover performance of IPAs with parallelization would make interesting future work, since multicore and GPU systems are commonly used for ZK proof generation. The short proof size, efficient verifier time and transparent setup make Bulletproofs appealing, and concretely faster prover implementations increase the size range of circuits that can be proven.

Size	**Our Impl.**	**[dVYA23]+parallel**	**[dVYA23]**	**verify**
2k	260	385 (1.5×)	2148 (8.3×)	19.94
4k	490	724 (1.5×)	4150 (8.5×)	15.51
8k	430	1358 (3.2×)	7963 (18.5×)	29.31
16k	872	2593 (3.0×)	15257 (17.5×)	50.78

Table 3: IPA proof generation and verification times in milliseconds (ms). Slowdown factors are relative to our method. The **[dVYA23]** implementation uses our parallelized MSM implementation and **[dVYA23]+parallel** additionally parallelizes the loop that updates generators.

B Circuits for fixed-base scalar multiplication

The scalar multiplication in or SNARK circuit for Π_{NR} has a fixed public base, so we investigated the potential of using optimized scalar multiplication but were not able to make it work efficiently. Here are some of the challenges.

- When computing G^x, if G is fixed and public, we can compute multiples $G^2, G^3, \ldots, G^{2^w}$ for a window size w and then speed up the simple double-and-add algorithm by expressing x in base w then doubling w times followed by adding one point from our precomputed table. We compute one doubling per bit of x, but can reduce the number of adds significantly, e.g., when $w = 8$ and $|x| = 256$ we have only 32 adds. In software having a small table of precomputed points, and looking them up is relatively cheap, since point arithmetic is by comparison expensive.

- In our circuit, point arithmetic is *not expensive:* checking doubles and adds each cost 3 constraints, and because of our choice of curve, all arithmetic is native. So if we save $w - 1$ adds per chunk of x, we need to lookup and add a point in fewer than $3(w - 1)$ constraints. When $w = 8$ we need to use no more than 21 constraints to come out ahead, but the lookup circuit we have uses 30 constraints. Even with free lookups, the number of constraints would only go down by about 768.

- In a related setting, [WOS+25] have a look-up mechanism (with interactive-R1CS) and also exploit fixed-base scalar multiplication when the curve operations are non-native. (With non-native operations each curve operation requires thousands of constraints, instead of 3)

- In [FW24] they use the "Baby Jub Jub" curve[3], for which there exists a circuit for fixed-based scalar multiplication that uses only 770 R1CS constraints. This is a savings of 1899 constraints vs. our current circuit (2669 constraints). The Baby jub jub circuit is written in Circom (part of `circomlib` [4])

  Using this circuit would require our scheme to change curves, and port the Circom implementation to bellpepper. This is a considerable effort, and might be an interesting future effort.

  The curve is defined over the scalar field of BN254, the pairing-friendly BN curve used in Ethereum. To keep our work in the prime-order setting we'd need a prime-order ordinary curve with this same order to use for BP and Spartan. Such a curve should be easy to find, but won't have the optimizations of the Pasta curves. So some of the performance gain of having a smaller circuit will be offset by having to run the proof system on a slower curve.

[3] https://docs.iden3.io/publications/pdfs/Baby-Jubjub.pdf

[4] https://github.com/iden3/circomlib/blob/35e54ea21da3e8762557234298dbb553c175ea8d/circuits/escalarmulfix.circom#L235