Vision: A Modular Framework for Anonymous Credential Systems

Anja Lehmann, Andrey Sidorenko, and Alexandros Zacharakis

Hasso Plattner Institute, University of Potsdam
{anja.lehmann,andrey.sidorenko,alexandros.zacharakis}@hpi.de

Abstract. Anonymous credentials enable the unlinkable presentation of previously attested information, or even only predicates thereof. They are a versatile tool and currently enjoy attention in various real-world applications, ranging from the European Digital Identity project to Privacy Pass. While each application usually requires their own tailored variant of anonymous credentials, they all share the same common blueprint. So far, this has not been leveraged though, and currently several proposals either targeting monolithic variants of core components such as BBS signatures, or application-specific protocols undergo standardization. This is clearly not optimal, as the same work gets repeated multiple times, while still risking ending up with many slight modifications of the same main idea and protocols. In this work we present our vision to use a modular approach to build anonymous credential systems: they are built from a core component – consisting of a commitment, signature and NIZK scheme – that can be extended with additional commitment-based modules in a plug-and-play manner. We sketch modules for pseudonyms, range proofs and device binding. Importantly, apart from the committed input, all modules are entirely independent of each other. We use this modularity to propose a concrete instantiation that uses BBS signatures for the core component and ECDSA signatures for device binding, addressing the need to bind modern credential schemes to legacy signatures in secure hardware elements.

1 Introduction

Anonymous credentials (AC) are the privacy-preserving version of classical digital certificates. They enable users to obtain attested attributes from a trusted issuer that they can later present towards a third-party verifier. On top of the standard security properties provided by classic certificates, anonymous credentials have several important privacy features built-in, such as selective disclosure of the attested attributes and multi-show unlinkability. Importantly, they guarantee these privacy properties even against corrupt issuers. This makes anonymous credentials the perfect tool for all applications that target or require privacy-preserving authentication – at least in theory. In the real-world however, anonymous credentials are still not widely deployed, despite major applications demanding exactly their capabilities. Most prominently, the European efforts

to build a European Digital Identity Wallet (EUDI) must adhere with strong privacy requirements stated in the eIDAS regulation, which explicitly demands that the system must support pseudonyms, selective disclosure and unlinkable authentications [Eurnd,eid14]. Similarly, Privacy Pass [DIW24], was invented to provide privacy-preserving rate-limiting for online services. Both require multi-show unlinkable presentation of certified information, for which anonymous credentials would have been a natural fit. However, neither application uses ACs, instead they reverted to batch-issuance of one-time tokens. As batch-issuance inherently incurs significant costs and challenges for secure deployment, the interest in "upgrading" to an AC-based solution remains strong.

Lack of Existing Standards. A core reason that hindered real-world adoption of anonymous credentials so far is the lack of suitable standards for the underlying cryptography. Apart from the ISO standards in the context of Direct Anonymous Attestation (DAA) [ISO13], no fully approved standard of these credential schemes exists. The DAA standard is partly outdated, and targets a particular application, which does not fit the requirements e.g., in the EUDI or Privacy Pass. The standard from 2013 already contains a protocol relying on BBS signatures [CL04] which is also a widely considered candidate for ACs today. While the core BBS signature has not changed since then, the associated zero-knowledge (ZKP) proof has seen significant efficiency improvements in the last decade [CDL16,TZ23,LKWL25].

Challenges in Standardization. The increased interest in deployment of anonymous credentials has already led to several standardization activities: BBS-based credentials currently undergo standardization in the IETF CFRG [LKWL25], including two extensions to support pseudonyms [KB25a] and blind issuance [KB25b]. Concrete protocols for using anonymous credentials in the context of Privacy Pass have been proposed in the respective IETF working group by Apple [YW25] and Google [SK25]. Further, a standard on Sigma protocols is being developed too, essentially abstracting the ZKP component therein [Orr25b]. ISO has also initiated a working draft on a new standard for Attribute-Based Credentials [ISO25].

As these standards have significant overlaps in the underlying techniques, this has already sparked discussions how to best package and combine these efforts [Pat25b]. Standards tailored to a specific application will not only repeat the same work, but might result in various incarnations of the same core concept with minor modifications – resulting in incompatible implementations and duplicated work across projects. Standards that focus on building blocks avoid such a lock-in situation on the application side, but in turn can be too shielded to be extendable and usable. This is in fact a new situation compared to more conventional cryptographic building blocks such as encryption or classic signatures. These classic primitives have monolithic APIs which can be composed on pure in/output behavior. In contrast, anonymous credentials are built from a core signature primitive upon which various statements via a ZKP are proven.

While the core part is relatively stable across application, the ZKP part and the exact feature set will vary greatly.

1.1 Modular Framework for Standardization

In this work we formulate our vision to standardize and build anonymous credential systems in a modular manner. We identify re-usable components – which we call *modules* – and design them in a way that makes them compatible via a core module, and allow for flexible extensions. Protocols can be built in a plug-and play manner from these modules, and standardization and security analysis can focus on modules instead of the entire credential system. By abstracting modules we also aim to support a crypto agile design where individual parts can have different instantiations that can be combined and replaced with ease.

Sign, Commit & Prove. Our framework abstracts the common blueprint that underlies *all* anonymous credential systems so far: the core component is a (multi-message) signature scheme used by the issuer to attest the user's attributes. The signature scheme must enable efficient non-interactive ZKPs (NIZK) of knowledge of such a signature on both *revealed* and *committed* attributes (all other attributes remain *hidden*). The commitments serve as the main glue between the main module and further extensions. That is, we define the core module (CD) to be a multi-message signature scheme which allows for such *committed disclosure proofs*. Any capability that goes beyond plain selective disclosure is added through modules that operate on the committed attributes. To exemplify the role of these modules, we sketch how *pseudonyms* and *range proofs* can be expressed therein. Notably, all our modules can come with their individual NIZK proof system, as they prove each module in an isolated way. This allows to use optimized NIZKs for different parts of the system, instead of aiming for a monolithic system that covers all. Further, security can be analyzed on the level of the independent modules and then propagates to the full system.

Explicit Device Binding. A particular focus in our modular framework is on *device-binding*, which is currently not covered by any of the standardization efforts, but is a crucial requirement in the EUDI context [Eurnd]. Device-binding aims to provide non-transferability of credentials, i.e., prevent malicious users from copying and sharing their credentials. It assumes that the user's host device includes a secure hardware component– the Secure Element (SE) – that contains cryptographic keys that cannot be extracted and are only accessible through a minimal signature API. The user's credential then includes a device public key as an attribute, and every presentation must include a fresh proof-of-possession (PoP) under that key.

We propose an *explicit device binding* solution that essentially mimics this standard approach in a privacy-preserving way: the SE is still assumed to only output a standard signature under a public key attested in the user's credential. The public key will only be revealed in committed form through the CD module, and the device-binding module adds a NIZK that proves knowledge of a fresh

PoP under that key. Notably, the proof system and signature scheme for this PoP can both be different from the main multi-signature scheme used for the attribute credential.

This approach differs from existing proposals in anonymous credential schemes that design these capabilities directly within the system itself [ISO13,CDL16,HSS23]. These solutions are highly-efficient, but require non-standard APIs from the SEs, typically involving computations over a pairing-friendly group. The reality is much more bleak though, and current SEs merely support ECDSA signatures over P-256 curves. As the development and lifecycle of SEs is extremely static, our explicit device binding enables to add non-transferability to anonymous credential systems in a way that is compatible with these hardware limitations. In fact, we believe that decoupling the aspect of device binding from the core credential component is essential for any short-term deployment. While signature schemes for embedded hardware devices will be notoriously hard – or even impossible – to update, signature schemes for the main credential itself do not adhere to such constraints. Our modular approach with explicit device binding caters for these differences, and does not impose that hardware limitations stemming from the SEs are propagated to the entire credential system.

Instantiation: BBS-ECDSA. As a concrete instantiation, we show how to use BBS signatures [BBS04,LKWL25] as the multi-message signature while relying on ECDSA for device binding. The NIZK of a BBS signature on committed attributes can be done through simple and lightweight Schnorr proofs, and using Pedersen commitments to bridge to further modules. For computing the PoP NIZK, we employ the Schnorr-based protocol ZKAttest/CDLS [FLM21,CLR24], extended with an additional proof to bridge the underlying elliptic curves, which differ between BBS and ECDSA. This BBS-ECDSA solution has been proposed and implemented by Ubique [Amr25] in the context of an innovation challenge of the German EUDI project, and was further improved by Dock Labs [Har25]. We believe this BBS-ECDSA combination to be of independent interest: it provides the simplicity and flexibility of BBS credentials, while supporting device-binding with legacy hardware. We provide an overview of this construction and demonstrate how it fits our proposed framework.

Related Work. We stress that none of the building blocks used in our framework are novel, and existing implementations may be used for large parts of the system. The modular approach relying on commitments to bridge and compose different instantiations is not new either, and has been used by Camenisch et al. [CKL+16] in the ABC4Trust project [ABC14]. Their framework focused on bridging the two prevalent constructions back in 2015, IBM's Identity Mixer and Microsoft's U-Prove, and showed how they can be combined to yield the same overall credential system. Here we do not believe that *the* credential system exists, but every application will require their own combination of features and modules. Further, our work fully decouples the modules from each other,

supporting different proof systems per module which we leverage to capture legacy-compliant device binding.

More generally, our framework follows the Commit-and-Prove approach [Kil90,CLOS02] that underlies many cryptographic protocols. The work by Bobolz et al. [BDK24] is also somewhat similar in spirit as it considers signatures on committed messages as main component, and shows how this yields provably secure anonymous credentials with selective disclosure and group signatures. Our work targets more expressive functionality, but does not aim at a formal study (yet).

Similar to us, various works also rely on commitments as linking component that allows to extend functionality and link different proofs from (potentially) different proving systems. García-Rodríguez et al. [GRKBS24] use Pedersen commitments as a bridging component to extend a Pointcheval-Sanders multi-message signature [PS16,PS18] instantiation with additional functionalities. Crescent [PPZ24] functions on a similar spirit, using Groth16 [Gro16] to prove knowledge of (legacy) credentials and outputting Pedersen commitments to attributes which can be fed to other proving systems to prove additional statements. In fact, Crescent can perfectly fit into the design paradigm described in this work. Finally, the implementation of DockNetwork [Har25] uses commitments to link information from different presentations and prove arbitrary statements on the attributes therein, the credential itself can be signed using BBS.

On the theory side, Camenisch et al. [CDHK15] utilize a modular design to construct a full-fledged AC system by composing multiple instances of a single-issuer AC system in the UC-framework. While that work also strives to formalize AC systems, our approach makes the key observation that the module selection of a single-issuer AC system may vary, and these modules may be independent of the proof system, thus bridging the gap towards legacy signature schemes, or highly optimized bespoke proof systems.

Vision & Future Work. The idea is to evolve this work in a community driven fashion. We invite the wider community to propose alternative instantiations for the modules presented here and also extend the considered module set – and of course exercise the framework through designing concrete applications from it. On the theoretical side, security models and the study of provably secure composition are obvious next steps. Overall, we hope our work can serve as a foundation for a common architecture for building and using modern anonymous credentials.

2 Preliminaries

Notation. We denote with λ the security parameter. To write the set $\{1, \ldots, m\}$, we use the shorthand $[m]$. For a set S, we denote with S^ℓ its ℓ-fold Cartesian product, namely $S^\ell = \{s_1, \ldots, s_\ell \mid s_j \in S\}$. For vectors, we use bold fonts. Given a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ we denote with $\boldsymbol{v}[i]$ the element v_i and with $\boldsymbol{v}[S]$ the

vector $(v_i)_{i \in S}$. We also denote $x \xleftarrow{\$} S$ to denote sampling an element from S uniformly at random and assigning it to x. For a group $\mathbb{G}$, denote the identity element in $\mathbb{G}$ as $1_{\mathbb{G}}$. We will use various groups based on Elliptic Curves. We associate each elliptic curve with two fields: the base field, where the elliptic curve point coordinates live, and the scalar field, where scalars lives.

Public (Shared) Parameters. We next define the basic cryptographic primitives we will use in this work. Since some of them will share public parameters, we assume there exists a global Setup algorithm that on input the security parameter 1^λ outputs the public parameters $pp(\lambda)$ (we will omit λ in what follows and simply write pp). These global public parameters are then given as input to instantiate the primitives. In our instantiations, such parameters will include group descriptions and hash functions.

(Multi-message) Signature Scheme. A signature scheme allows a party in possession of a secret key to digitally sign a message m.

Definition 1 ((Multi-message) Signature Scheme). *A signature scheme* Sig *is a tuple of algorithms*

$\mathsf{Sig.Setup}(pp) \to pp_{\mathsf{Sig}}$*: probabilistically generates parameters* pp_{Sig} *from* pp*,*
$\mathsf{Sig.KeyGen}(pp_{\mathsf{Sig}}) \to (sk, pk)$*: probabilistically generates a key pair* (sk, pk)*,*
$\mathsf{Sig.Sign}(sk, m) \to \sigma$*: a probabilistic algorithm that takes as input a secret key* sk *and a message* m[1] *and outputs a signature* σ*,*
$\mathsf{Sig.Vf}(pk, m, \sigma) \to \{0, 1\}$*: a deterministic algorithm that takes as input a public key* pk*, a message* m *and a signature* σ *and outputs a bit.*

When the message space is a vector of ℓ *elements, namely* $\mathcal{M}^\ell$*, we call the scheme a* ***multi-message*** *signature scheme* MSig*.*

A signature Sig is secure when it satisfies the standard existential unforgeability under chosen message attacks (EUCMA). We define the notion formally in Appendix A.2 .

Commitment Schemes. A commitment scheme Com allows a prover to commit to a message that she can later reveal.

Definition 2 (Commitment Scheme). *A commitment scheme with canonical opening* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit})$ *consists of two algorithms:*

$\mathsf{Com.Setup}(pp) \to pp_{\mathsf{Com}}$ *: probabilistically generates parameters* pp_{Com} *from* pp*,*
$\mathsf{Com.Commit}(pp_{\mathsf{Com}}, m; \rho) \to C$ *: probabilistically outputs a commitment* C *from parameters* pp_{Com} *and a message* m *using the explicit randomness* ρ*.*

Verification is canonical, i.e. to verify a commitment C to a message m given the randomness ρ, the verifier asserts that $C \stackrel{?}{=} \mathsf{Com.Commit}(pp_{\mathsf{Com}}, m; \rho)$.

The security guarantees are (1) *binding*, i.e. a commitment can only be opened to a unique message and (2) *hiding*, i.e. the commitment does not reveal any information about the underlying message.

[1] The message m belongs to some message space $\mathcal{M}$ implicit in pp_{Sig}.

Non-Interactive Zero-Knowledge Proofs. A Non-Interactive Zero-Knowledge NIZK proof for an NP relation $\mathcal{R}$ allows a prover to compute a proof for the validity of a statement x.

Definition 3 (Non-Interactive Zero-Knowledge Proofs). *A Non-Interactive Zero-Knowledge (NIZK) proof system for an NP relation $\mathcal{R}$ is a tuple of efficient algorithms:*

$\mathsf{NIZK.Setup}(pp) \to pp_{\mathcal{R}}$**:** *A probabilistic algorithm that on input the global parameters pp outputs parameters $pp_{\mathcal{R}}$.*

$\mathsf{NIZK.P}(pp_{\mathcal{R}}, x, w) \to \pi$**:** *A probabilistic algorithm that on input the parameters $pp_{\mathcal{R}}$ and a statement/witness pair (x, w) belonging to $\mathcal{R}$ outputs a proof π.*

$\mathsf{NIZK.V}(pp_{\mathcal{R}}, x, \pi) \to \{0, 1\}$ *A deterministic algorithm that on input the parameters $pp_{\mathcal{R}}$, the statement x and a proof π outputs a bit.*

It satisfies completeness, stating that an honest prover will always succeed in verifying an accepting proof; knowledge soundness, stating that a convincing proof implies knowledge of a valid witness w; zero knowledge, stating that a proof for statement x reveals nothing but the validity of the statement (i.e. there exists some w s.t. $(x, w) \in \mathcal{R}$). We give a more formal definition in Appendix A.2 .

We use the Camenisch-Stadler notation [CS97] to succinctly present the relation to be proven for a NIZK. To denote, e.g., the discrete logarithm relation, we write:

$$\underbrace{DL}_{\text{relation}} = \left\{ \underbrace{(y)}_{\text{witness}} : \underbrace{Y = y \cdot G}_{\text{statement}} \right\}$$

Here, G and Y are public values that constitute the statement and y is the witness. Unless otherwise stated, we assume all variables in the statement, which are not part of the witness, to be public, or at least known to the verifier.

In our constructions, we usually need to associate proofs with some context ctx (e.g. some random nonce sent by the verifier). This corresponds to a signature of knowledge [CL06]. Since all the NIZK used in this work can be turned into signatures of knowledge by including the context in the Fiat-Shamir input, we simply abuse notation and write $\mathsf{P}(pp_{\mathcal{R}}, x, w)[ctx]$ and $\mathsf{V}(pp_{\mathcal{R}}, x, \pi)[ctx]$ when we want to make the context explicit.

3 Modular Framework for Anonymous Credentials

In this section we present our core anonymous credential construction that supports *modular functionality* via the commit-and-prove approach [Kil90,CLOS02]. We start by presenting an informal description of the setting and the security properties we aim to achieve and then describe our black box construction. To this end, we first identify the core relation underlying any presentation which we call *committed disclosure*; roughly, this allows proving ownership of a credential while revealing some attributes and committing to some others. We then explain how to achieve modularity by proving statements over the committed attributes. To demonstrate this, we present the black-box construction with one (w.l.o.g.)

generic relation $\mathcal{R}$ over the committed attributes, modelling an abstract module. We then give concrete examples of modules, focusing on usual use cases and requirements of anonymous credentials. Concretely, we show how to specialize the construction with modules for proving (1) possession of a device key, (2) range proofs and (3) pseudonyms.

Anonymous Credentials: Functionality & Properties. There are three entities in anonymous credentials: the *issuers*, the *users* and the *verifiers*. Issuers attest attributes in form of a credential to the user. If a user wants to authenticate towards a verifier, she derives a presentation from her credential that reveals the minimal amount of information in a verifiable manner. The presentation convinces the verifier that the presented information is authentic.

We require that the system satisfies *unlinkability*, *credential unforgeability* and *presentation unforgeability*. Informally,

Unlinkability. Captures user privacy needs; it asserts that presentations cannot be traced back to a user and that different presentations of the same credential cannot be linked even when corrupted issuers and verifier collude,

Credential unforgeability. Captures that only the issuer can create valid credentials.

Presentation unforgeability. Captures that during presentation only information attested by the issuer can be accepted, and optionally that the attributes of the credential satisfy additional properties.

Note that presentation unforgeability allows extending the "basic" presentation requirement (committed disclosure) of the system. In particular, we can define one or more relations that capture additional system functionalities. We can then implement a module for each functionality and add it on top of the core construction. The latter property should require that during presentation the additional relations are satisfied by the credential attributes.

We emphasize that we only provide an informal treatment of security and leave defining a formal security model and security proofs as future work.

3.1 Commit-and-Prove Construction

We now present the idea of our modular framework, that yields an anonymous credential scheme.

We present our core black box construction w.r.t. some generic relation $\mathcal{R}$ abstracting the additional functionality added via modules. The core idea is to use a multi-message signature scheme MSig to sign a set of attributes $\boldsymbol{a} \in \mathcal{M}^{\ell}$ and derive the credential *cred*. During presentation, the prover commits to some attributes (for which it wants to prove additional statements) and reveals some other. Then it proves that it owns a credential that is consistent with the revealed attributes and the commitments (i.e. the credential attributes are the openings of the commitments). We identify the *committed disclosure relation* (CD) that captures the above requirements. Having established a set of commitments whose

openings are verifiably part of the credential, we can prove an *additional relation* on the committed values captured by the relation $\mathcal{R}$ in a commit-and-prove fashion. To this end, we employ a NIZK for $\mathcal{R}$. The commitment scheme acts as the glue between the two statements: since in both proofs we present statements about the committed values and since the user can only possibly know a single opening, the proven statement for $\mathcal{R}$ must hold for the attributes contained on the credential *cred*.

The committed disclosure relation. We start by defining the committed disclosure relation CD. Let MSig[2] and Com be a multi-message signature scheme with message space $\mathcal{M}^\ell$ and a commitment scheme[3] with message space $\mathcal{M}$, and two (disjoint) subsets $\mathsf{S_{Rev}}, \mathsf{S_{Com}} \subseteq [\ell]$ corresponding to the sets of revealed and committed attributes. These sets also implicitly define the set of hidden attributes $\mathsf{S_{Hid}} = [\ell] \backslash (\mathsf{S_{Rev}} \cup \mathsf{S_{Com}})$. We define the relation CD as:

$$\mathsf{CD}\left\{\left(\begin{array}{c}\boldsymbol{a} \in \mathcal{M}^\ell, cred, \\ \{\rho_i\}_{i\in\mathsf{S_{Com}}}\end{array}\right) : \begin{array}{l}\mathsf{MSig.Vf}(ipk, \boldsymbol{a}, cred) = 1 \\ \forall i \in \mathsf{S_{Rev}} : a_{\mathsf{Rev},i} = \boldsymbol{a}[i] \\ \forall i \in \mathsf{S_{Com}} : C_i = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \boldsymbol{a}[i]; \rho_i)\end{array}\right\}$$

This relation underpins all reasonable AC constructions in the sense that it (1) asserts possession of a credential issued by the issuer due to the $\mathsf{MSig.Vf}$ condition and (2) discloses some of the attributes in a presentation. We also assume an efficient $\mathsf{NIZK_{CD}}$ for the relation CD.

Additional functionality captured by relations. We next explain how to express additional functionalities by expressing them as NP relations. Let Φ be a predicate capturing the additional functionality that should hold during a presentation. Φ involves some of the credential attributes and possibly additional public and secret inputs x, w (looking ahead, this can for example correspond to a valid signature over some committed public key). We define $\mathcal{R}$ as the relation over the committed attributes that shows that (1) the predicate Φ is satisfied and (2) the inputs to the predicate correspond to the commitments' openings. Formally we have:

$$\mathcal{R}\left\{\left(\boldsymbol{a} \in \mathcal{M}^{|\mathsf{S}_\mathcal{R}|}, \{\rho_i\}_{i\in\mathsf{S}_\mathcal{R}}, w\right) : \begin{array}{l}\Phi(\boldsymbol{a}, w, x) \\ \forall i \in \mathsf{S}_\mathcal{R} : C_i = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \boldsymbol{a}[i]; \rho_i)\end{array}\right\}$$

We also assume a $\mathsf{NIZK}_\mathcal{R}$ for proving statements of the relation $\mathcal{R}$[4] and rely on its properties to prove the added module secure.

[2] Aligning with the context of ACs, we refer to a signature of MSig as *cred*.

[3] For ease of presentation, we use one commitment per committed attribute. One can modify the construction to support compact commitments that will give a single commitment for multiple attributes.

[4] One might want to relax the zero knowledge property of this NIZK and allow partial leakage of w (see Remark 1).

Our Construction. We now explain the construction along the four phases of a credential scheme: Setup, IssueCred, Present, and Verify. We present the construction assuming an additional abstract module captured by a relation $\mathcal{R}$. The core building blocks are a multi-message signature MSig scheme with message space $\mathcal{M}^\ell$, a commitment scheme Com with message space $\mathcal{M}$, as well as two NIZK systems, $\mathsf{NIZK}_{\mathsf{CD}}, \mathsf{NIZK}_{\mathcal{R}}$ for proving statements for CD and $\mathcal{R}$ respectively.

- Setup. The system setup involves generating global parameters $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and running each primitive's $\mathsf{Setup}(pp)$ algorithm to specialize them, namely $pp_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(pp)$, $pp_{\mathsf{CD}} \leftarrow \mathsf{NIZK}_{\mathsf{CD}}.\mathsf{Setup}(pp)$, $pp_{\mathcal{R}} \leftarrow \mathsf{NIZK}_{\mathcal{R}}.\mathsf{Setup}(pp)$. The issuer key generation involves running the multi-message signature scheme's KeyGen to get a key pair, namely, $(isk, ipk) \leftarrow \mathsf{MSig.KeyGen}(pp_{\mathsf{MSig}})$.
- IssueCred. Issuance is a protocol run between the issuer and user, at the end of which the user obtains her attested attributes $\boldsymbol{a}$ in form of an issuer-signed credential *cred*. We assume the simplest form of issuance here, and let the issuer sign all attributes through the multi-message signature scheme; concretely, the issuer generates the credential as $cred \leftarrow \mathsf{MSig.Sign}(isk, \boldsymbol{a})$. Extending this to blind issuance is an interesting future extension, in particular as blind issuance is necessary whenever native device binding is needed, where one of the attributes is a device secret key that is not known to the user. Note however that blind issuance is not needed for the explicit device binding feature we introduce later, as we realize device binding through an explicit binding, where the credential contains the device public key, not the secret key (see discussion of native vs. explicit device binding below).
- Present. The user starts by preparing some attributes $\boldsymbol{a}_{\mathsf{Rev}}$ to be revealed (corresponding to an index set $\mathsf{S}_{\mathsf{Rev}}$) and computes fresh commitments to the set of attributes $\boldsymbol{a}_{\mathsf{Com}}$ that need to be committed (corresponding to index set $\mathsf{S}_{\mathsf{Com}}$). She then uses $\mathsf{NIZK}_{\mathsf{CD}}.\mathsf{P}$ to assert these are parts of a credential *cred* under the issuer's key *ipk*. Finally, she invokes $\mathsf{NIZK}_{\mathcal{R}}.\mathsf{P}$ to compute a proof about the statement $\mathcal{R}$ involving a subset of $\boldsymbol{a}_{\mathsf{Com}}$ corresponding to an index set $\mathsf{S}_{\mathcal{R}} \subseteq \mathsf{S}_{\mathsf{Com}}$ and possibly additional statement/witness x, w. For the latter, she uses the same commitments as in the proof of CD. The presentation proof consists of the commitments $\{C_i\}_{i \in \mathsf{S}_{\mathsf{Com}}}$ and the two NIZK proofs $\pi_{\mathsf{CD}}, \pi_{\mathcal{R}}$.
- Verify. The verifier takes the presentation proof $\pi = (\{C_i\}_{i \in \mathsf{S}_{\mathsf{Com}}}, \pi_{\mathsf{CD}}, \pi_{\mathcal{R}})$ and invokes the $\mathsf{NIZK}_{\mathsf{CD}}.\mathsf{V}$ for verifying π_{CD} using these commitments; it then takes the subset of commitments $\{C_i\}_{i \in \mathsf{S}_{\mathcal{R}}}$ and invokes $\mathsf{NIZK}_{\mathcal{R}}.\mathsf{V}$ with these commitments (and additional input x). It accepts if both proof are accepting.

We present the algorithms for Present, Verify in Figure 1. We emphasize that each relation can have their own optimized NIZK proof, as they are generated and verified in a modular way, only proving their respective sub-statement. This yields a crypto agile and modular design, and in particular caters for the fact that we might have to prove statements over different cryptographic primitives – each having their optimized NIZK.

Remark 1 (Relaxed NIZK). We can relax the properties of the $\mathsf{NIZK}_{\mathcal{R}}$ to achieve better efficiency. In particular, we can relax the perfect/statistical zero knowledge requirement by allowing some "leakage" of the witness w (while still requiring perfect zero knowledge for the $\boldsymbol{a}$ part of the witness which is vital for unlinkability). As an example, in the construction for proof of possession of an ECDSA signature we present in Section 4.1, we note that since this is used as a one-time signature, parts of the signature can be revealed without compromising privacy (if the Secure Element is assumed to be trusted) while providing better overall efficiency for the system. We refer the reader to [LÖ25,FFK+23] for more details on how to define such relaxed NIZKs and leave the systematic study and analysis of this generalization as future work.

$\underline{\mathsf{Present}(ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \mathsf{S}_{\mathcal{R}}, \boldsymbol{a}, cred, x, w, ctx_V)}$

Compute commitments $\forall i \in \mathsf{S}_{\mathsf{Com}}$: $\mathsf{Com.Commit}(pp_{\mathsf{Com}}, a_i; \rho_i)$
Prepare statements/witnesses for relations CD, $\mathcal{R}$:

$$x_{\mathsf{CD}} \leftarrow (ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \{a_i\}_{i\in\mathsf{S}_{\mathsf{Rev}}}, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}), \quad w_{\mathsf{CD}} \leftarrow (\boldsymbol{a}, cred, \{\rho_i\}_{i\in\mathsf{S}_{\mathsf{Rev}}})$$

$$x_{\mathcal{R}} \leftarrow (\mathsf{S}_{\mathcal{R}}, \{C_i\}_{i\in\mathsf{S}_{\mathcal{R}}}, x), \quad w_{\mathcal{R}} \leftarrow (\{a_i, \rho_i\}_{i\in\mathsf{S}_{\mathcal{R}}}, w)$$

Compute $ctx \leftarrow ctx_V||C_1||\cdots||C_\ell$ and output $\pi = (\{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, \pi_{\mathsf{CD}}, \pi_{\mathcal{R}})$ where

$$\pi_{\mathsf{CD}} \leftarrow \mathsf{NIZK}_{\mathsf{CD}}.\mathsf{P}(pp_{\mathsf{CD}}, x_{\mathsf{CD}}, w_{\mathsf{CD}})[ctx], \quad \pi_{\mathcal{R}} \leftarrow \mathsf{NIZK}_{\mathcal{R}}.\mathsf{P}(pp_{\mathcal{R}}, x_{\mathcal{R}}, w_{\mathcal{R}})[ctx]$$

$\underline{\mathsf{Verify}(ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \mathsf{S}_{\mathcal{R}}, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, \{a_{\mathsf{Rev},}\}_{i\in\mathsf{S}_{\mathsf{Rev}}} x, ctx_V, \pi)}$

Parse $\pi = (\{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, \pi_{\mathsf{CD}}, \pi_{\mathcal{R}})$ and prepare the statements for relations CD, $\mathcal{R}$:

$$x_{\mathsf{CD}} \leftarrow (ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \{a_{\mathsf{Rev},i}\}_{i\in\mathsf{S}_{\mathsf{Rev}}}, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}), \quad x_{\mathcal{R}} \leftarrow (\mathsf{S}_{\mathcal{R}}, \{C_i\}_{i\in\mathsf{S}_{\mathcal{R}}}, x),$$

Compute $ctx \leftarrow ctx_V||C_1||\cdots||C_\ell$ and Output $b_{\mathsf{CD}} \wedge b_{\mathcal{R}} \wedge b_{\mathsf{S}}$ where

$$b_{\mathsf{S}} \leftarrow \mathsf{S}_{\mathcal{R}} \stackrel{?}{\subseteq} \mathsf{S}_{\mathsf{Com}} \wedge \mathsf{S}_{\mathsf{Com}} \stackrel{?}{\subseteq} [\ell] \wedge \mathsf{S}_{\mathsf{Rev}} \stackrel{?}{\subseteq} [\ell] \wedge \mathsf{S}_{\mathsf{Com}} \cap \mathsf{S}_{\mathsf{Rev}} \stackrel{?}{=} \varnothing$$

$$b_{\mathsf{CD}} \leftarrow \mathsf{NIZK}_{\mathsf{CD}}.\mathsf{V}(pp_{\mathsf{CD}}, x_{\mathsf{CD}}, \pi_{\mathsf{CD}})[ctx], \quad b_{\mathcal{R}} \leftarrow \mathsf{NIZK}_{\mathcal{R}}.\mathsf{V}(pp_{\mathcal{R}}, x_{\mathcal{R}}, \pi_{\mathcal{R}})[ctx]$$

Fig. 1: The Present and Verify parts of a credential presentation.

Instantiating MSig *and* CD. Natural candidates for the MSig are the BBS and PS signature schemes. Both schemes are build for that purpose, support multi-messages, are efficient, and well understood. Furthermore, they natively support efficient NIZKs for committed disclosure when combined with the Pedersen commitment scheme through standard Schnorr protocols, yielding a highly-efficient instantiation for the core module. Another approach is to use ECDSA for the MSig in combination with a zkSNARK (e.g. [Fs24,PPZ24,WOS+25]); this has

the advantage of supporting legacy credentials, at the cost of an increased complexity for the NIZK. When post-quantum security is needed, one could use a post-quantum signature scheme such as ML-DSA in combination with a lattice based zkSNARK (e.g. [BS23]) and a hash based commitment scheme; or one of the recent lattice-based multi-message signature schemes [BLNS23,DKLW25] with a simpler NIZK for committed disclosure.

Remark 2 (Supporting KVAC). We state our framework for publicly-verifiable credentials using multi-message signatures, but this can be easily turned into a variant for keyed-verification anonymous credentials (KVACs) too. KVACs can be seen as the symmetric version of ACs, that were developed to avoid the need for pairing-friendly curves that are required for all efficient multi-message signatures. By assuming the issuer and verifier to be the same entity, and turning verification into a keyed operation, anonymous credential schemes can be run on standard elliptic curves. Essentially, the pairing check in the verification is replaced through a computation using the issuer secret key. Apart from this check, KVACs and ACs are typically equivalent, and, e.g. MAC versions for both BBS and PS exist [CMZ14,Orr24,DDKT25]. KVACs will only differ in this core module, and can re-use the same extension modules, e.g., for range proofs or pseudonyms.

3.2 Device Binding Module

In the digital identity context, an essential additional functionality is *device binding*, which we can express through an appropriate relation in our framework. This property aims to ensure non-transferability of credentials. To this end, the credential gets bound to a cryptographic device key that is protected through a hardware component, e.g., a Secure Element (SE). The hardware protection ensures that the key cannot be extracted, and any presentation of the credential requires access to that device.

In particular, we consider the case where the user owns a device that stores a secret key of a signature scheme and provides an API to request signatures on messages that are valid under the corresponding public key. The user's credential contains the device public key as an attribute, and credential presentation requires to also prove knowledge of a device signature under that attested (but not revealed) device public key on a random challenge nonce sent by the verifier.

Explicit vs. Native Device Binding. We focus on such *explicit* device binding that essentially follows the classic proof-of-possession paradigm used for conventional certificates. The main benefit of that approach is modularity, as it fully decouples the device part from the main credential and NIZK thereof. In particular, it enables to combine different schemes for the main attribute credential and the device signature, which we exemplify in Section 4 with the BBS-ECDSA scheme. An alternative approach is to rely on *native* device-binding which is directly built into the credential scheme. The Direct Anonymous Attestation (DAA) protocol [ISO13,CDL16] is such a variant. Therein, the device *secret* key

is embedded through a blind issuance protocol as an attribute in the main credential, and the NIZK is then jointly computed by the device and host. Such native device binding will result in much better performance, both in computational costs and proof sizes, but requires dedicated interfaces from the SE for the credential scheme. Currently, deployed SEs do not provide interfaces that enable native device binding for BBS though, which is why we focus on the explicit device binding option here.

High-Level Idea. We next describe how explicit device binding works. We assume a secure element that stores internally a device secret key dsk of a signature scheme Sig and exposes an API to produce signatures under this key, namely, when queried on some message m, it outputs $\sigma \leftarrow \mathsf{Sig.Sign}(sk, m)$. The user presenting the credential gets some challenge nonce and queries its secure element to produce $\sigma \leftarrow \mathsf{Sig.Sign}(sk, \mathsf{nonce})$. She is then charged to prove knowledge of the (secret) signature σ under the (secret) public key encoded in the attributes.

To achieve this, the user encodes its device public key as a set of attributes which are included in the credential during issuance. During the presentation, the user includes these attributes to the set $\mathsf{S}_{\mathsf{Com}}$ when defining the committed disclosure statement. The task now is reduced to proving knowledge of a signature σ on nonce under the *committed encoding of the device public key*. This can be done with a NIZK for an appropriate relation $\mathcal{R}$ capturing *proof of possession*.

More formally, let $\mathsf{Encode}(pk) \rightarrow m \in \mathcal{M}^t$ be an efficient algorithm that allows encoding a public key of Sig as a vector of elements in $\mathcal{M}^t$ for a fixed t, where $\mathcal{M}$ is the space where attributes live. The user needs to prove the following relation:

$$\mathsf{PoP}\left\{\left(\{dpk_i, \rho_i\}_{i\in[t]}, \sigma\right) : \begin{array}{l} \mathsf{Sig.Vf}(dpk, \mathsf{nonce}, \sigma) = 1 \\ \mathsf{Encode}(dpk) = \mathbf{dpk} \\ \forall i \in [t]\ C_i = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, dpk_i; \rho_i) \end{array}\right\}$$

The corresponding predicate Φ in the proof of possession case captures that a vector $\mathbf{dpk}$ encodes some public key dpk and a (secret) signature σ on some (public) message nonce verifies under this key. The credential must contain the encoding $\mathbf{dpk}$ as dedicated attributes.

Instantiating PoP*.* Due to restrictions imposed by the available secure elements, smartphone-based applications are currently limited to using ECDSA over the $\mathsf{P\text{-}256}$ curve. We present in Section 4.1 an efficient instantiation for $\mathsf{NIZK}_{\mathsf{PoP}}$ with ECDSA based on existing ideas when the core module is implemented via BBS. We note, however, that in principle any signature scheme could be used when such hardware supports it. A scheme "compatible" with the MSig of the core module would translate to significant efficiency improvements.

3.3 Additional Modules: Range Proofs & Pseudonyms

We next describe the relations and possible instantiations of additional modules to the base construction. In particular, we consider the range proof module,

allowing to prove that attributes lie in a range, and the pseudonym module, allowing to deterministically derive pseudonyms from the credential, that yield a form of controlled linkability.

Range Proofs. A typical feature of ACs is allowing to prove that an attribute a lies in some range, namely $l \leqslant a < h$. This can be used for example to achieve credential revocation, proof of non-expiration, or pseudonyms as we will see next. We can add such functionality by simply extending the system with the range proof relation defined next:

$$\mathsf{RP}\left\{(a,\rho) : l \leqslant a < h,\ C = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, a; \rho)\right\}$$

In this case there is no additional secret input and the public input consists of the bounds l, h. We also note that we can run different instances of this relation when multiple attributes need to satisfy some range.

Instantiating RP. Bulletproofs can be easily applied to instantiate range proofs [BCC+16,BBB+18]. They provide a very small proof size, require no trusted setup, and have been extensively analyzed in academic literature. It is also well-known in the community how to construct zero-knowledge range proofs from polynomial commitments, e.g. as summarized in [BFGW20].

Additionally, there is also an ongoing discussion within the IETF, both at the CFRG [Pat25a], and the Privacy Pass working group [Gol25] about the inclusion of such range proofs in standards. One particular scheme considered in the discussions is [KRS25]. Detailing and refining a concrete instantiation for RP is an open next step here, too.

Pseudonyms. Pseudonyms allow to introduce a level of controlled linkability. They are identifiers (deterministically) derived from a dedicated high-entropy seed *pns* embedded in the user's credential. Of particular interest are so called scope-exclusive pseudonyms [ABC14] that ensure that only a unique pseudonym can be derived per scope *scp*. Pseudonyms across scopes are still fully unlinkable. If the scope encodes the identity of a service, this enables pseudonymous authentication while still guaranteeing that only a single account per user can be generated. A more flexible variant considers a looser upper bound *lim* and ensures that a user can create up to *lim* unlinkable pseudonyms per scope. The latter is currently considered for rate-limiting in the anonymous credential variant of Privacy Pass [YW25].

All existing constructions for such pseudonyms can be expressed through a PRF as $nym = \mathsf{PRF}(pns, scp||ctr)$. The PRF key is an attested high-entropy seed *pns* in the main credential that must never get revealed, whereas the scope is public. The counter *ctr* can either be revealed, or just proven to be below a limit *lim*. We state the more general version and again start from a committed input for *pns*:

$$\mathsf{NYM}\left\{(pns, \rho, ctr) : \begin{array}{l} nym = \mathsf{PRF}(pns, scp||ctr),\ 1 \leqslant ctr < lim \\ C = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, pns; \rho) \end{array}\right\}$$

Instantiating NYM. The simplest way to instantiate the PRF is to use HashDH leading to $nym = pns \cdot \mathsf{H}(scp||ctr)$, and using a simple Schnorr proof for $\mathsf{NIZK}_{\mathsf{NYM}}$. This pseudonym generation (w/o counter) is standardized as part of DAA by ISO [ISO13] and undergoes standardization in the IETF for BBS credentials [KB25a], and the counter-based variant is suggested for rate-limited credentials based on CMZ MAC [YW25]. This construction reveals the counter though, which reduces the anonymity set with every presentation. If the counter should remain hidden, one can use the Dodis-Yampolskiy PRF instead [CHK+06], which was recently proposed for rate-limited BBS credentials [SK25]. The drawback of both constructions is that their privacy relies on the DL assumption, which would make the entire credential system susceptible to harvest now – link latter attacks. A construction for pseudonyms that is compatible with DL-based credential schemes, yet has everlasting (and thus PQ) privacy was recently proposed by Chairattana-Apirom et al. [CADLT25]. All aforementioned solutions require Schnorr proofs for their NIZKs, and thus are very lightweight. An alternative for post-quantum privacy is to simply rely on AES as PRF, for which highly-optimized zkSNARKs exist [OKMZ25].

3.4 Discussion of Modularity & Instantiation Choices

Our core motivation is to build anonymous credential schemes in a more modular way, to simplify both the theoretical analysis as well as their implementations and standardization efforts. The price we pay is a slight overhead through the commitment layer: every functionality beyond the core CD is bridged over a commitment and the opening is possibly proven repeatedly in every module. Thus, on paper, it will be slightly less efficient than a targeted construction where the entire statement is proven through a monolithic NIZK. However we believe that this is a reasonable trade-off, and the added costs of a few group elements will not matter in most applications. Further, if many schemes reuse the same components, this will likely lead to optimized implementations that can compensate for these additional steps. In fact, we believe that this modularity will rather lead to better efficiency than a monolithic solution for the following reasons.

Optimal Signatures for CD *vs* PoP. Our device binding approach leverages the observation that the signature schemes for the main credential and the device *do not need to be the same*, which can lead to both better efficiency and simpler systems (and therefore easier to standardize). Concretely, our task is reduced to optimizing NIZKs for CD and PoP. The only limitation is that the NIZKs used for both must be compatible with the commitment scheme Com[5]. Depending on the application requirements and trade-offs, one can use an appropriate instantiation (for example zkSNARKs to optimize proof size or sigma protocols to optimize proving time or even combinations of them).

[5] In fact, this limitation is even weaker since one can use different commitment schemes and additionally prove they commit to the same values.

MSig	$\mathsf{NIZK}_{\mathsf{CD}}$		Sig	$\mathsf{NIZK}_{\mathsf{PoP}}$
BBS	Schnorr proof		ECDSA	Schnorr proof (CDLS)
PS	Schnorr proof		ECDSA	zkSNARK, e.g.
ECDSA	zkSNARK (various)	+		[Fs24,PPZ24,WOS+25]
PQC-MSig	native NIZK [LNP22]		Schnorr	Schnorr proof
[BLNS23,DKLW25]			ML-DSA	LaBRADOR [BS23]
ML-DSA	LaBRADOR [BS23]			

Fig. 2: Examples for MSig, Sig, and proof primitive combinations.

This separation also allows to address hardware or legacy limitation in parts of the system, without imposing this on all modules. As mentioned, device binding is currently restricted to using ECDSA. However, we do have flexibility on choosing the MSig of the issuer for which dedicated schemes such as BBS or PS provide significant advances in terms of simplicity and efficiency. The same design pattern can be applied when migrating to post-quantum solutions. While future device binding will most likely again be restricted to standard signatures, i.e., ML-DSA, the multi-message signature schemes can rather be an optimized variant that natively supports NIZKs, e.g., [BLNS23,DKLW25]. In fact, in the transition time even mixed settings are plausible, where only the main credential should be quantum-safe, but device binding is still done via ECDSA[6].

In Figure 2 we present a table with possibilities for instantiating anonymous credentials with device binding support. Not listed here is the commitment construction, which, of course, can be instantiated as needed. All the possible choices for a building block "suite" for a credential can be compared regarding their efficiency, quantum-readiness, proof size, and other aspects.

Taming the Complexity of zkSNARKs. Our modular approach can also help to address a significant barrier for zkSNARK-based solutions: it is notoriously complex to properly design secure "circuits" for concrete computations. This affects also standardization; in fact, despite significant efforts, it seems we are still far from having standards for zkSNARKs (see for example the current status in [ZKP24]). With our modular approach, this complexity can be partially alleviated since one would now need to design and optimize smaller and simpler circuits. For example, one could adapt "full-fledged" zkSNARK solutions to prove knowledge of ECDSA-based credentials such as [Fs24,PPZ24,WOS+25] to merely instantiate the core and device binding modules, essentially translating

[6] Note that this makes sense for two reasons. First, assuming a high cost to quantum-break a signature scheme, the consequences of doing that on the issuer level is catastrophic compared to compromising individual users' keys. Second, whereas issuer signatures can be replaced upon a quantum-threat, upgrading secure elements in already shipped phones appears impossible. Thus, for device binding, ECDSA might still be the only available option during such a transition time.

the ECDSA-attested information into a Pedersen commitment. Then, for all further relations that need to be proven on the user's attributes one can use simple Schnorr proofs.

4 BBS Credentials with ECDSA Device Binding

Here we present an instantiation of our black box construction with proof-of-possession functionality based on modules from Figure 2, specifically BBS as the multi-message signature and ECDSA (over P-256) for device binding. This is based on the BBS-ECDSA idea that has been proposed and implemented by Ubique [Amr25]. To demonstrate the full AC system based on our module choices above, we also provide a concrete example for anonymous age verification proofs in Appendix D. More concretely, we instantiate the primitives of Section 3 as follows:

MSig. We choose BBS for implementing the issuer side MSig, since it is the most mature multi-message signature scheme, with native support for efficient selective disclosure (which can be easily fine-tuned for committed disclosure). It is also currently undergoing standardization [LKWL25], and we can use the main operations $\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf}$ from this draft without any modifications. BBS requires a pairing-friendly group with $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q, e.g., BLS12-381 [7].

$\mathsf{NIZK}_{\mathsf{CD}}$. To prove the committed disclosure relation, BBS natively supports efficient NIZK proofs via Schnorr proofs [TZ23]. Note that here our core module *deviates from the BBS draft* which only considers selective disclosure, whereas we need to also prove equivalence over committed inputs. Including this capability requires little effort, and yields a more versatile BBS standard.

Sig. We use ECDSA for the PoP signature scheme. While ECDSA can work with any group $\mathbb{G}$, secure element support is currently limited to NIST P-256. Crucially, we use a group different from the one of BBS.

Com. We use the Pedersen commitments Ped over the group $\mathbb{G}_1$ of BBS.

$\mathsf{NIZK}_{\mathsf{PoP}}$. We use a combination of a sigma protocol based approach using the techniques of [FLM21,CLR24] for proving knowledge of a signature in combination with techniques from [COPZ22,OKMZ25] for showing that two Pedersen commitments over different groups admit the same opening.

We formally state the standard building blocks for ECDSA and Pedersen commitments in Appendix A.3 . In the remainder of this section we focus on the most challenging part which is bridging the ECDSA signature to a BBS credential.

4.1 Proof-of-Possession of ECDSA Signature

The main challenge towards an efficient implementation of a NIZK for ECDSA proof-of-possession is the need for foreign field arithmetic, that is, proving statements over a field different than the one the statements are about. In particular,

[7] Other curves, such as those from [AFG24] may be used, in particular when a higher security level is required.

verifying an ECDSA signature is a computation over $\mathbb{Z}_t$, the base field of P-256 while the committed public key lives in $\mathbb{Z}_q$, the scalar field of BLS12-381.

Our approach to mitigate the efficiency issues that rise from this is to "transfer" the committed public key to a different curve, amenable to efficient computations over $\mathbb{Z}_t$. Concretely, the prover first presents *fresh* commitments to the (encoded) device public key dpk over a curve with scalar field $\mathbb{Z}_t$. She then shows that the new commitment indeed encodes the same public key as the one presented in the committed attributes. Finally, she proves knowledge of a signature of a challenge nonce over the *committed* public key. Note that the last step is now a native computation. This corresponds to expressing the relation PoP as a conjunction of two relations as follows:

$$\mathsf{ECDSA}\left\{\left(dpk, \sigma, \{\rho_i\}_{i\in[t]}\right) : \begin{array}{c}\mathsf{ECDSA.Vf}(dpk, \mathsf{nonce}, \sigma) = 1 \\ \forall i \in [t]\ C_i = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, dpk_i; \rho_i) \\ \mathsf{Encode}(dpk) = (dpk_1, \ldots, dpk_t)\end{array}\right\}$$

$$\mathsf{EQ}_t\left\{\left(\{dpk_i, \rho_i, \rho'_i\}_{i\in[t]}\right) : \forall i \in [t] \quad \begin{array}{l} C_i = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, dpk_i; \rho_i) \\ C'_i = \mathsf{Ped}_q.\mathsf{Commit}(pp_q, dpk_i; \rho'_i)\end{array}\right\}$$

In the above, we use the subscripts t, q for the Pedersen commitments over $\mathbb{Z}_t$ (the base field of P-256) and $\mathbb{Z}_q$ (the scalar field of BLS12-381) respectively. We also treat the witness elements dpk_i as integers. Note that while similar, the relation for ECDSA is different from the generic PoP relation since it involves a commitment scheme for a different domain. The EQ_t relation is used to show consistency between the two sets of commitments.

Next, we give an overview of an instantiation for each of the above proofs based on sigma protocols. We emphasize however that *any* instantiation works; one could for example use SNARKs for proving the ECDSA relation with smaller proof size and slower proving time[8].

Tom Curve. As mentioned, our proof of knowledge of an ECDSA signature needs to be defined as a computation over $\mathbb{Z}_t$ – the base field of P-256. When working on elliptic curves, however, the native computation is defined over the *scalar field* of the curve. We therefore need an elliptic curve whose scalar field is the base field of P-256. Such a curve is the Tom curve T-256 introduced in [FLM21]. Our proof for the relation ECDSA will be defined over this curve and the proof EQ will show that pairs of commitments over T-256 and BLS12-381 respectively open to the same value (as integers). Therefore, our full construction involves three curves: P-256, T-256 and BLS12-381. To avoid confusion about field notation we present in Table 1 the symbols we use for the fields of each curve (when needed).

We also use the notation Ped_q with parameters pp_q to denote the Pedersen commitment over the curve with scalar field $\mathbb{Z}_q$ (and similarly for $\mathbb{Z}_p, \mathbb{Z}_t$). Finally, given a point $P = (x, y)$ we denote its affine coordinates as $(x, y) = (P.x, P.y)$.

[8] There are some limitation on the families of SNARKs that can be (natively) used due to the properties of the field $\mathbb{Z}_t$. In particular, one should use a SNARK that is efficient in curves that are not "FFT-friendly".

Curve	Scalar Field	Base Field
P-256	$\mathbb{Z}_p$	$\mathbb{Z}_t$
T-256	$\mathbb{Z}_t$	
BLS12-381	$\mathbb{Z}_q$	

Table 1: Notation for base and scalar fields for the Elliptic Curves used in the manuscript.

PoKs for equality across different groups. To prove that the opening of two commitments over different groups are equal, we employ the techniques of [COPZ22,OKMZ25]. There, a simple protocol is presented for the relation EQ_1.

Note the "promise" $x < 2^b$. Due to technical reasons, the protocol requires a bound on the committed value x. The concrete choice of the value b affects the efficiency metrics of the protocol[9]. We must therefore handle the two following issues:

1. the coordinates of the ECDSA key *dpk* are longer than the supported bound,
2. the verifier must be convinced that the promise is satisfied.

For the first, we can do a "limb decomposition" of x as suggested in [COPZ22,OKMZ25]. Concretely, we decompose x as $x = \sum x_i 2^{ib}$ with $x_i \in [0, 2^b)$. We then run a proof of equality for all limbs x_i. The second issue is handled during issuance. In particular, the user presents to the issuer the coordinates of the point corresponding to the ECDSA public key and its encoding in 64-bit limbs and the issuer asserts that (1) it corresponds to a valid point and (2) the limbs lie in the correct range. Since the promise is guaranteed to hold a priori, to implement the EQ_8 proof, we simply provide the 8 proofs for the statement EQ of each limb. We present the construction in Appendix C.1.

PoKs for ECDSA *Verification.* Having transferred the commitments to T-256, proving knowledge of an ECDSA signature is now (mostly) a native computation. To create a proof, we follow the techniques of [FLM21,CLR24]. Roughly, we can reduce the verification of a signature to two claims: (1) proving that three committed (in affine form) points satisfy $P_1 + P_2 = P_3$ over P-256, and (2) proving that a scalar multiplication equation of the form $P_1 = sP_2$ holds where the points and the scalar are committed in T-256 and P-256 respectively. The former is a native computation and can be proven with standard sigma protocols. The latter is more involved due the two fields involved; nevertheless sigma protocol techniques can still be applied, albeit with some additional efficiency overhead. We also take advantage of the fact that the signed nonce is public and that

[9] We will use the bound $b = 64$ and therefore a point $Q = (x, y)$ can be represented as 8 64-bit values, 4 for each of the coordinates. This is used by current implementations. We expect, however, that using 2 limbs of 128 bits and encoding only coordinate x would result in a more efficient construction.

parts of the signature can be revealed without compromising user privacy (if the secure element is assumed to trusted). We present the details in Appendix C.2 .

Putting it all together. We next give a high level overview of the NIZK for PoP by combining the above building blocks. We encode the ECDSA public key $dpk = (x, y) \in$ P-256 as 8 attributes corresponding to 64-bit limbs of the coordinates x, y. These are included in the BBS credential *cred*. To make a presentation, the user starts by providing commitments to these attributes *over both* BLS12-381 and T-256 and uses $\mathsf{NIZK}_{\mathsf{EQ}}$ to create proofs $\{\pi_{x,i}, \pi_{y,i}\}_{i=1}^{4}$ that they correspond to the same value. Then, she creates two T-256 commitments C_x, C_y to the coordinates (x, y) by homomorphically combining the corresponding commitments and proves that she knows a valid signature on nonce w.r.t. to the committed key C_x, C_y. The latter is done by invoking $\mathsf{NIZK}_{\mathsf{ECDSA}}$. We defer details to Appendix C.2 . We plan to provide a full description of the approach and formally analyze its security in future work.

Acknowledgments. We thank Patrick Amrein and Fabian Aggeler from Ubique for many fruitful discussions on their BBS-ECDSA idea that has been captured in this work. We further thank Andrea Flamini for helpful feedback on an earlier draft of this work. This research was partially funded by the HPI Research School on Systems Design and by SPRIND, the Federal Agency for Breakthrough Innovation. The authors are solely responsible for the content of this publication; the positions presented here do not reflect the views of SPRIND.

References

ABC14. EU FP7 Trust & Security Program ABC4TRUST. Attribute-based credentials for trust. *ABC4TRUST project. Available online at https://abc4trust.eu/*, 2014.

AFG24. Diego F. Aranha, Georgios Fotiadis, and Aurore Guillevic. A short-list of pairing-friendly curves resistant to the special TNFS algorithm at the 192-bit security level. *CiC*, 1(3):3, 2024.

Amr25. Patrick Amrein. https://github.com/UbiqueInnovation/zkattest-rs, 2025.

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. IEEE Computer Society Press, May 2018.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Berlin, Heidelberg, August 2004.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Berlin, Heidelberg, May 2016.

BDK24. Jan Bobolz, Jesus Diaz, and Markulf Kohlweiss. Foundations of anonymous signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Jeremy Clark and Elaine Shi, editors, *FC 2024, Part II*, volume 14745 of *LNCS*, pages 121–139. Springer, Cham, March 2024.

BFGW20. Dan Boneh, Ben Fisch, Ariel Gabizon, and Zac Williamson. A simple range proof from polynomial commitments. Technical report, February 2020. Accessed: 2025-09-12.

BLNS23. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 384–417. Springer, Cham, August 2023.

BS23. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548. Springer, Cham, August 2023.

CADLT25. Rutchathon Chairattana-Apirom, Nico Döttling, Anna Lysyanskaya, and Stefano Tessaro. Everlasting anonymous rate-limited tokens. Cryptology ePrint Archive, Paper 2025/1030, 2025.

CDHK15. Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Berlin, Heidelberg, November / December 2015.

CDL16. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9*, pages 1–20. Springer, 2016.

CHK+06. Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 201–210, New York, NY, USA, 2006. Association for Computing Machinery.

CKL+16. Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 3–24. Springer, Cham, August 2016.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Berlin, Heidelberg, August 2004.

CL06. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Berlin, Heidelberg, August 2006.

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

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*, volume 14861 of *LNCS*, pages 69–93. Springer, Cham, July 2024.

CMZ14. Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.

Com25. European Commission. European age verification solution: Overall architecture and technical specifications. https://ageverification.dev/av-doc-technical-specification/docs/architecture-and-technical-specifications/, 2025.

COPZ22. Melissa Chase, Michele Orrù, Trevor Perrin, and Greg Zaverucha. Proofs of discrete logarithm equality across groups. Cryptology ePrint Archive, Report 2022/1593, 2022.

CS97. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424. Springer, Berlin, Heidelberg, August 1997.

DDKT25. Nicolas Desmoulins, Antoine Dumanois, Seyni Kane, and Jacques Traoré. Making BBS anonymous credentials eIDAS 2.0 compliant. https://eprint.iacr.org/2025/619, 2025.

DIW24. Alex Davidson, Jana Iyengar, and Christopher A. Wood. The Privacy Pass Architecture. RFC 9576, June 2024.

DKLW25. Adrien Dubois, Michael KlooSS, Russell W. F. Lai, and Ivy K. Y. Woo. Lattice-based proof-friendly signatures from vanishing short integer solutions. Cryptology ePrint Archive, Paper 2025/356, 2025.

eid14. Regulation (eu) 2024/1183 of the european parliament and of the council of 11 april 2024 amending regulation (eu) no 910/2014 as regards establishing the european digital identity framework. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910, July 2014. Accessed: 2024-10-18.

Eurnd. European Commission. Eu digital identity wallet home. https://ec.europa.eu/digital-building-blocks/sites/spaces/EUDIGITALIDENTITYWALLET/pages/694487738/EU+Digital+Identity+Wallet+Home, n.d. Accessed: 2025-10-07.

FFK+23. Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zk-SNARKs. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part III*, volume 14371 of *LNCS*, pages 455–485. Springer, Cham, November / December 2023.

FLM21. Armando Faz-Hernández, Watson Ladd, and Deepak Maram. ZkAttest: Ring and group signatures on top of existing ECDSA keys. Cryptology ePrint Archive, Report 2021/1183, 2021.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

Fs24. Matteo Frigo and abhi shelat. Anonymous credentials from ECDSA. Cryptology ePrint Archive, Report 2024/2010, 2024.

Gol25. Ian Goldberg. [privacy-pass] re: New protocol for rate-limited tokens. https://mailarchive.ietf.org/arch/msg/privacy-pass/xK8bi2p08w5k3JjXzoQLUn6YXIo/, March 2025. Mailing list post, accessed March 13, 2026.

GRKBS24. Jesús García-Rodríguez, Stephan Krenn, Jorge Bernal Bernabe, and Antonio Skarmeta. Beyond selective disclosure: Extending distributed p-abc implementations by commit-and-prove techniques. *Computer Networks*, 248:110498, 2024.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.

Har25. Lovesh Harchandani. hhttps://github.com/docknetwork/crypto/tree/main/equality_across_groups, 2025.

HSS23. Julia Hesse, Nitin Singh, and Alessandro Sorniotti. How to bind anonymous credentials to humans. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3047–3064, Anaheim, CA, August 2023. USENIX Association.

ISO13. ISO. Iso 20008-2:2013: Information technology - security techniques - anonymous digital signatures - part 2: Mechanisms using a group public key. Standard, International Organization for Standardization, Geneva, CH, November 2013.

ISO25. Iso/iec wd 24843: Information security - attribute-based credentials. Working Draft ISO/IEC WD 24843, International Organization for Standardization, 2025. Under development. A working group has prepared a draft.

KB25a. Vasilis Kalos and Greg M. Bernstein. Bbs per verifier linkability. Internet Draft draft-irtf-cfrg-bbs-per-verifier-linkability-02, Internet Research Task Force, September 2025.

KB25b. Vasilis Kalos and Greg M. Bernstein. Blind bbs signatures. Internet Draft draft-irtf-cfrg-bbs-blind-signatures-01, Internet Research Task Force, March 2025.

Kil90. Joe Kilian. *Uses of randomness in algorithms and protocols*. MIT Press, 1990.

KRS25. Jonathan Katz, Mariana Raykova, and Samuel Schlesinger. Anonymous credentials with range proofs and rate limiting. https://docs.rs/crate/authenticated-pseudonyms/latest/source/design/Range.pdf, February 2025. Accessed: 2025-09-12.

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, July 2025. Work in Progress.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.

LÖ25. Anja Lehmann and Cavit Özbay. Commit-and-prove system for vectors and applications to threshold signing. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part III*, volume 15676 of *LNCS*, pages 200–232. Springer, Cham, May 2025.

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

Orr24. Michele Orrù. Revisiting keyed-verification anonymous credentials. Cryptology ePrint Archive, Report 2024/1552, 2024.

Orr25a. Michele Orrù. Fiat-Shamir Transformation. Internet-Draft draft-irtf-cfrg-fiat-shamir-00, Internet Engineering Task Force, August 2025. Work in Progress.

Orr25b. Michele Orrù. Sigma protocols. Internet Draft draft-orru-zkproof-sigma-protocols-00, February 2025.

Pat25a. Christopher Patton. [cfrg] re: Incorporating or proofs in sigma protocol draft. https://mailarchive.ietf.org/arch/msg/cfrg/dF3t3_VY6eAeohjLyN5SbxP2NNI/, September 2025. Mailing list post, accessed March 13, 2026.

Pat25b. Christopher Patton. [privacy-pass] arc, acs: Dividing the work with cfrg. https://mailarchive.ietf.org/arch/msg/privacy-pass/AgLCkR1I_DdP1Q9eSALbDGmkkVM/, August 2025. Mailing list post, accessed March 13, 2026.

PPZ24. Christian Paquin, Guru-Vamsi Policharla, and Greg Zaverucha. Crescent: Stronger privacy for existing credentials. Cryptology ePrint Archive, Report 2024/2013, 2024.

PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Cham, February / March 2016.

PS18. David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338. Springer, Cham, April 2018.

SK25. Samuel Schlesinger and Jonathan Katz. Anonymous Credit Tokens. Internet-Draft draft-schlesinger-cfrg-act-00, Internet Engineering Task Force, August 2025. Work in Progress.

TZ23. Stefano Tessaro and Chenzhi Zhu. Revisiting bbs signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 691–721. Springer, 2023.

WOS+25. Anna P. Y. Woo, Alex Ozdemir, Chad Sharp, Thomas Pornin, and Paul Grubbs. Efficient proofs of possession for legacy signatures. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 3291–3308. IEEE Computer Society Press, May 2025.

WTs+18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

YW25. Cathie Yun and Christopher A. Wood. Anonymous Rate-Limited Credentials. Internet-Draft draft-yun-cfrg-arc-01, Internet Engineering Task Force, August 2025. Work in Progress.

ZKP24. ZKProof Standards Committee. Plan for 2024/2025 - zkproof standards. https://docs.zkproof.org/standards#plan-for-20242025, 2024. Accessed: 2025-10-10.

A Additional Preliminaries

Here we present the formal definitions of all omitted building blocks and their security notions.

A.1 Deferred Definitions

An adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm. We write $y \leftarrow \mathcal{A}(x)$ to denote output y of $\mathcal{A}$ on input y.

Definition 4 (Bilinear Groups). *Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of prime order q, where $\mathbb{G}_1$ and $\mathbb{G}_2$ use additive and $\mathbb{G}_T$ uses multiplicative notation. For generators G_1 of $\mathbb{G}_1$ and G_2 of $\mathbb{G}_2$, we define $\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ to be a pairing (bilinear map) if $\mathbf{e}$ can be computed* efficiently *and fulfills the following conditions [CL04]:*

1. *Bilinearity: $\forall a, b \in \mathbb{Z}_q\colon \mathbf{e}(a \cdot G_1, b \cdot G_2) = \mathbf{e}(G_1, G_2)^{ab}$.*
2. *Non-degeneracy: $\mathbf{e}(G_1, G_2)$ generates G_T.*

We specifically consider type-3 pairings, for which no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known.

Definition 5 (Group Generator (resp. Bilinear Group Generator)). *A group (resp. bilinear) generator algorithm is an efficient algorithm $\mathcal{G}$ (resp. $\mathcal{BG}$), that, given a security parameter 1^λ as input, outputs the description $(q, \mathbb{G}, G)$ of a group $\mathbb{G}$ of prime order q with generator G (resp. $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathbf{e})$ of a bilinear group, where $\mathbf{e}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing, G_1 generates $\mathbb{G}_1$, G_2 generates $\mathbb{G}_2$, and q is a prime such that $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$). The order q has a bit-length polynomial in λ.*

Definition 6 (Pseudorandom Function). *A pseudorandom function is an efficient function $\mathsf{PRF} : K \times X \to Y$. Intuitively, for a random, secret key $k \xleftarrow{\$} K$, the pseudorandom outputs of $\mathsf{PRF}(k, x)$ are indistinguishable from a random function for $X \to Y$.*

A.2 Formal Security Properties

Existential Unforgeability. A signature scheme Sig is EUCMA-secure, if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}^{\mathsf{Sig.Sign}(sk,\cdot)}$ with access to a signing oracle,

$$\Pr\left[\begin{array}{c|c} \mathsf{Sig.Vf}(pk, m, \sigma) = 1 & pp_{\mathsf{Sig}} \leftarrow \mathsf{Sig.Setup}(pp) \\ \text{and} & (sk, pk) \leftarrow \mathsf{Sig.KeyGen}(pp_{\mathsf{Sig}}) \\ m \notin \mathcal{Q} & (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sig.Sign}(sk,\cdot)}(pk) \end{array}\right] \leqslant \mathsf{negl}(\lambda)$$

where $\mathcal{Q}$ denotes the set of queries of $\mathcal{A}$ to the signing oracle. EUCMA security for MSig is defined analogously, requiring the forged message to differ in at least one component from all previously signed message vectors:

$$\Pr\left[\begin{array}{c|c} \mathsf{MSig.Vf}(pk, \boldsymbol{m}, cred) = 1 & pp_{\mathsf{Sig}} \leftarrow \mathsf{MSig.Setup}(pp) \\ \text{and} & (sk, pk) \leftarrow \mathsf{MSig.KeyGen}(pp_{\mathsf{Sig}}) \\ \boldsymbol{m} \notin \mathcal{Q} & (\boldsymbol{m}, cred) \leftarrow \mathcal{A}^{\mathsf{MSig.Sign}(sk,\cdot)}(pk) \end{array}\right] \leqslant \mathsf{negl}(\lambda)$$

Commitment security

Definition 7. *A commitment scheme must for all λ and universal parameters $pp(\lambda)$ satisfy the binding and hiding properties as defined next:*

(Computational) Binding: *For all PPT $\mathcal{A}$, the following should hold:*

$$\Pr\left[\begin{array}{c|c} C_0 = C_1 \\ \wedge m_0 \neq m_1 & \begin{array}{c} pp_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(pp) \\ m_0, m_1, \rho_0, \rho_1 \leftarrow \mathcal{A}(pp_{\mathsf{Com}}) \\ C_0 \leftarrow \mathsf{Com.Commit}(pp_{\mathsf{Com}}, m_0, \rho_0) \\ C_1 \leftarrow \mathsf{Com.Commit}(pp_{\mathsf{Com}}, m_1, \rho_1) \end{array} \end{array}\right] \leqslant \mathsf{negl}(\lambda)$$

(Perfect) Hiding: *For all λ, m_0, m_1, the distributions $\{\mathsf{Com.Commit}(pp_{\mathsf{Com}}, m_0)\}$ and $\{\mathsf{Com.Commit}(pp_{\mathsf{Com}}, m_1)\}$ are perfectly indistinguishable where $pp_{\mathsf{Com}} \leftarrow \mathsf{Com.Setup}(pp)$.*

Non-Interactive Zero-Knowledge Proofs

Definition 8 (NIZK security). *A Non-Interactive Zero-Knowledge (NIZK) proof system for an NP relation $\mathcal{R}$ satisfies the following properties for all $\lambda \in \mathbb{N}$, $pp(\lambda)$:*

Completeness: *For all $(x, w) \in \mathcal{R}$:*

$$\Pr\left[1 \leftarrow \mathsf{NIZK.V}(pp_{\mathcal{R}}, x, \pi) \middle| \begin{array}{c} pp_{\mathcal{R}} \leftarrow \mathsf{NIZK.Setup}(pp) \\ \pi \leftarrow \mathsf{NIZK.P}(pp_{\mathcal{R}}, x, w) \end{array}\right] = 1$$

Knowledge-Soundness: *For every PPT algorithm P^* there exists a PPT algorithm $\mathcal{E}$ such that*

$$\Pr\left[1 \leftarrow \mathsf{NIZK.V}(pp_{\mathcal{R}}, x, \pi) \wedge (x, w) \notin \mathcal{R} \middle| \begin{array}{c} pp_{\mathcal{R}} \leftarrow \mathsf{NIZK.Setup}(pp) \\ (x, \pi) \leftarrow \mathsf{P}^*(pp_{\mathcal{R}}; \rho) \\ w \leftarrow \mathcal{E}(pp_{\mathcal{R}}; \rho) \end{array}\right] \leqslant \mathsf{negl}(\lambda)$$

Zero-Knowledge: *There exist PPT algorithms $\mathcal{S}_1, \mathcal{S}_2$ such that for all $(x, w) \in \mathcal{R}$, the distributions*

$$\{pp_{\mathcal{R}}, \pi \mid pp_{\mathcal{R}} \leftarrow \mathcal{S}_1(pp), \pi \leftarrow \mathcal{S}_2(pp_{\mathcal{R}}, x)\}$$

$$\{pp_{\mathcal{R}}, \pi \mid pp_{\mathcal{R}} \leftarrow \mathsf{NIZK.Setup}(pp), \pi \leftarrow \mathsf{NIZK.P}(pp_{\mathcal{R}}, x, w)\}$$

are statistically indistinguishable.

Pseudorandomness

Definition 9 (Pseudorandomness of a PRF). *We call a PRF F pseudorandom, if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}^{f_b(\cdot)}$ with access to an evaluation oracle, the following holds:*

$$\Pr\left[b' = b \middle| \begin{array}{c} k \xleftarrow{\$} K, b \xleftarrow{\$} \{0,1\} \\ f_0(x) \xleftarrow{\$} Func, f_1(x) \leftarrow \mathsf{F}(k, x) \\ b' \leftarrow \mathcal{A}^{f_b(\cdot)}(1^\lambda) \end{array}\right] \leqslant \mathsf{negl}(\lambda)$$

where $Func$ is the set of all functions $X \to Y$.

A.3 Concrete Schemes

We now present concrete schemes for commitments and digital signatures. We assume that public parameters include the group description: $(\mathbb{G}, G, q) \in pp$.

Pedersen Commitments. The Pedersen commitment, where the committing party chooses $\rho \xleftarrow{\$} \mathbb{Z}_q$ is given in Figure 4. We use G' and H' to differentiate the parameters from the "regular" group with generator G. Pedersen commitments are also homomorphic, i.e. given two commitments $C_0 = m_0 \cdot G' + \rho_0 \cdot H'$ and $C_1 = m_1 \cdot G' + \rho_1 \cdot H'$, one can compute the commitment to $m_2 = m_0 + m_1$ as $C_2 = C_0 + C_1 = (m_0 + m_1) \cdot G' + (\rho_0 + \rho_1) \cdot H'$ for the opening $\rho_2 = \rho_0 + \rho_1$.

$\mathsf{Ped.Setup}(pp)$

Parse $(\mathbb{G}, q) \in pp$

$generator', H' \xleftarrow{\$} \mathbb{G}^2$

Output $pp_{\mathsf{Com}} \leftarrow (\mathbb{G}, G', q), H'$

$\mathsf{Ped.Commit}(pp_{\mathsf{Com}}, m \in \mathbb{Z}_q; \rho) \in \mathbb{Z}_q$

Output $C \leftarrow m \cdot G' + \rho \cdot H'$

Fig. 4: Ped commitments for a group with order q

ECDSA Signatures. The ECDSA digital signature scheme is given in Figure 3, assuming the group order is p (P-256 in our case). There also exists an alternative pair of signing/verification routines for ECDSA, as described in [FLM21], which are more efficient for zero-knowledge proofs. The alternative verification can also be found in Figure 3. Since we assume that secure elements produce standard signature, thus we omit the alternative signing method. A standard ECDSA signature $\sigma = (r, s)$ can be converted to a signature $\sigma' = (K, z)$ by computing K as in ECDSA.Vf and setting $z \leftarrow s \cdot r^{-1}$.

$\mathsf{ECDSA.Setup}(pp)$

Parse $(\mathbb{G}, G, p) \in pp$
Choose a random $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$
$F : \mathbb{G} \to \mathbb{Z}_p$ is the function that outputs the x-coordinate of the group element mod p
Output $pp_{\mathsf{ECDSA}} \leftarrow (\mathbb{G}, G, p), \mathsf{H}, F$

$\mathsf{ECDSA.KeyGen}(pp_{\mathsf{ECDSA}})$

$x \xleftarrow{\$} \mathbb{Z}_p$
$Q \leftarrow x \cdot G$
Output $sk \leftarrow x, pk \leftarrow Q$

$\mathsf{ECDSA.Sign}(pp_{\mathsf{ECDSA}}, sk \in \mathbb{Z}_p, m \in \{0,1\}^*)$

$k \xleftarrow{\$} \mathbb{Z}_p$
$K \leftarrow k \cdot G$
$r \leftarrow F(K)$
$s \leftarrow k^{-1}(\mathsf{H}(m) + r \cdot sk) \mod p$
Output $\sigma \leftarrow (r, s)$

$\mathsf{ECDSA.Vf}(pp_{\mathsf{ECDSA}}, pk \in \mathbb{G}, m \in \{0,1\}^*, \sigma \in \mathbb{Z}_p^2)$

Parse $(r, s) \leftarrow \sigma$
If $pk = 1_{\mathbb{G}} \vee r \notin \mathbb{Z}_p^* \vee s \notin \mathbb{Z}_p^*$, output 0
$u_1 \leftarrow \mathsf{H}(m) \cdot s^{-1}$, $u_2 \leftarrow r \cdot s^{-1}$
$K \leftarrow u_1 \cdot G + u_2 \cdot pk$
Output $F(K) \stackrel{?}{=} r$

$\mathsf{ECDSA.Convert}(pp_{\mathsf{ECDSA}}, pk \in \mathbb{G}, m \in \{0,1\}^*, \sigma \in \mathbb{Z}_p^2)$

Parse $(r, s) \leftarrow \sigma$
$u_1 \leftarrow \mathsf{H}(m) \cdot s^{-1}$, $u_2 \leftarrow r \cdot s^{-1}$
$K \leftarrow u_1 \cdot G + u_2 \cdot pk$
$z \leftarrow u_2^{-1}$
Output $\sigma' \leftarrow (K, z)$

$\mathsf{ECDSA.Vf'}(pp_{\mathsf{ECDSA}}, pk \in \mathbb{G}, m \in \{0,1\}^*, \sigma' \in \mathbb{G} \times \mathbb{Z}_p)$

Parse $(K, z) \leftarrow \sigma'$
If $pk = 1_{\mathbb{G}} \vee K = 1_{\mathbb{G}} \vee z \notin \mathbb{Z}_p^*$, output 0
$r \leftarrow F(K)$
Output $z \cdot K - \mathsf{H}(m) \cdot r^{-1} \cdot G \stackrel{?}{=} pk$

Fig. 3: The ECDSA signature scheme. We also include an alternative verification algorithm on a modified signature and a routine to convert the signature to the alternative form.

B Committed Disclosure with BBS

BBS [CL04] is a multi-message signature scheme, we define it formally in Figure 5. BBS has been studied extensively in academic literature. One of its main advantages is the compatibility with efficient and powerful native zero-knowledge proofs.

$\mathsf{BBS.PGen}_{\mathsf{BBS}}(pp)$

Parse $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathbf{e}) \in pp$
$H_1, \ldots, H_\ell \xleftarrow{\$} \mathbb{G}_1^\ell$
Output $pp_{\mathsf{BBS}} \leftarrow (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, \mathbf{e}, H_1, \ldots, H_\ell)$

$\mathsf{BBS.KeyGen}_{\mathsf{BBS}}(pp_{\mathsf{BBS}})$

$x \xleftarrow{\$} \mathbb{Z}_q$
$X_2 \leftarrow x \cdot G_2$
Output $sk \leftarrow x, pk \leftarrow X_2$

$\mathsf{BBS.Sign}_{\mathsf{BBS}}(pp_{\mathsf{BBS}}, sk \in \mathbb{Z}_q, (a_1, \ldots, a_\ell) \in \mathbb{Z}_q^\ell)$

$C((a_1, \ldots, a_\ell)) \leftarrow G_1 + \sum_{i=1}^{\ell}(a_i \cdot H_i)$
$e \xleftarrow{\$} \mathbb{Z}_q$
$A \leftarrow \frac{1}{e+x} \cdot C((a_1, \ldots, a_\ell))$
Output $cred \leftarrow (A, e)$

$\mathsf{BBS.Vf}_{\mathsf{BBS}}(pp_{\mathsf{BBS}}, pk \in \mathbb{G}_2, (A, e) \in \mathbb{G}_1 \times \mathbb{Z}_q, (a_1, \ldots, a_\ell) \in \mathbb{Z}_q^\ell)$

$C((a_1, \ldots, a_\ell)) \leftarrow G_1 + \sum_{i=1}^{\ell}(a_i \cdot H_i)$
Output $\mathbf{e}(A, X_2) \stackrel{?}{=} \mathbf{e}(-e \cdot A + C((a_1, \ldots, a_\ell)), G_2)$

Fig. 5: BBS signature scheme

To commit to attributes, we employ the well-known Pedersen commitment scheme, which we recall in Appendix A.3. We reuse the $\mathbb{G}_1$ group of BBS for Ped to make both schemes directly compatible. The parameters pp_{Com} contain two uniformly distributed generators $G', H' \in \mathbb{G}_1$ (independent of the ones used for BBS). Each attribute is committed to independently. For every presentation, the user computes new commitments for each $i \in \mathsf{S_{Com}}$:$\rho_i \xleftarrow{\$} \mathbb{Z}_q, C_i \leftarrow \mathsf{Ped.Commit}(pp_{\mathsf{Com}}, a_i; \rho_i)$ during Present (see also Figure 1). Each commitment is then formed as $C_i = a_i \cdot G' + \rho_i \cdot H'$.

Given a credential (A, e), the user can re-randomize it to achieve unlinkability, where the rerandomized values $\overline{A}, \overline{B}$, and D are revealed to verifier RP. The user then proves knowledge of (1) the hidden attributes of $\mathsf{S_{Hid}}$ and (2) the correspondence of committed attributes in C_i to the ones in the credential, as indicated by $\mathsf{S_{Com}}$. The user computes a proof of committed disclosure π_{CD} as

shown in Figure 6. Note that before the NIZK is executed the prover and verifier need to agree on the statement; in particular, the user sends fresh commitments C_i as described in Section 3.1.

$\underline{\mathsf{NIZK}_{\mathsf{CD}}.\mathsf{P}(pp_{\mathsf{CD}}, (ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, \{a_i\}_{i\in\mathsf{S}_{\mathsf{Rev}}}), (\boldsymbol{a}, \{\rho_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, (A, e)))[ctx]}$

Rerandomize BBS credential

$$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$$

$$\overline{A} \leftarrow (r_1/r_2) \cdot A \qquad D \leftarrow (1/r_2) \cdot C(\mathbf{a}) \qquad \overline{B} \leftarrow r_1 \cdot D - e \cdot \overline{A}$$

Prepare statement for remaining linear relations BBS_Σ,:

$$x_\Sigma \leftarrow (\overline{A}, \overline{B}, D, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}\mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}},), \quad w_\Sigma \leftarrow (e, r_1, r_2, \{a_i\}_{i\in[\ell]\setminus\mathsf{S}_{\mathsf{Rev}}}, \{\rho_i\}_{i\in\mathsf{S}_{\mathsf{Com}}})$$

Define the remaining linear relation

$$\mathsf{BBS}_\Sigma \left\{ (x_\Sigma, w_\Sigma) : \begin{array}{c} \overline{B} \leftarrow r_1 \cdot D - e \cdot \overline{A} \\ C(\mathbf{a}_{\mathsf{S}_{\mathsf{Rev}}}) = r_2 \cdot D - \sum_{i\in[\ell]\setminus\mathsf{S}_{\mathsf{Rev}}} (a_i \cdot H_i) \\ \forall i \in \mathsf{S}_{\mathsf{Com}} : a_i \cdot G' + \rho_i \cdot H' = C_i \end{array} \right\}$$

Generate proof

$$\pi_\Sigma \leftarrow \mathsf{NIZK}_{\mathsf{BBS}_\Sigma}.\mathsf{P}(pp_\Sigma, x_\Sigma, w_\Sigma)[ctx]$$

Output $\pi_{\mathsf{CD}} = (\overline{A}, D, \overline{B}, \pi_\Sigma)$

$\underline{\mathsf{NIZK}_{\mathsf{CD}}.\mathsf{V}(pp_{\mathsf{CD}}, (ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}, \{a_i\}_{i\in\mathsf{S}_{\mathsf{Rev}}}), \pi_{\mathsf{CD}})[ctx]}$

Parse $(\overline{A}, D, \overline{B}, \pi_\Sigma) \leftarrow \pi_{\mathsf{CD}}$
Prepare statement for linear relations BBS_Σ,:

$$x_\Sigma \leftarrow (\overline{A}, \overline{B}, D, \{C_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}\mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}},),$$

Output b_Σ where

$$b_\Sigma \leftarrow \mathsf{NIZK}_{\mathsf{BBS}_\Sigma}.\mathsf{V}(pp_\Sigma, x_\Sigma, \pi_\Sigma)[ctx]$$

Fig. 6: Non-Interactive zero-knowledge proof system $\mathsf{NIZK}_{\mathsf{CD}}$ for committed disclosure of a BBS credential.

B.1 Comparison with BBS IETF Draft

Core BBS Signature. Where applicable we show how the algorithms for BBS can be expressed in terms of several current IETF drafts. The core BBS credential description can be closely modeled by [LKWL25]:

- $\mathsf{PGen}_{\mathsf{BBS}}$ corresponds to running KeyGen on random bytes and then $SkToPk$ on the secret key (Sec. 3.4).
- $\mathsf{Sign}_{\mathsf{BBS}}$ corresponds to running Sign (Sec. 3.5.1).

– $\mathsf{Vf}_{\mathsf{BBS}}$ is equivalent to running Verify (Sec. 3.5.2) by rearranging the equation. The exact verification in [LKWL25] is the following:

$$\mathbf{e}(A, X_2) \cdot \mathbf{e}(e \cdot A + C(\mathbf{a}), G_2) \stackrel{?}{=} \mathbf{e}(G_1, G_2).$$

Zero-knowledge Proofs. The core difference to [LKWL25] lies in the ZKP: [LKWL25] "hard-codes" a specific proof for selective disclosure on a rerandomized BBS signature. This is not the case in our scheme, as an additional proof of equivalence between the signed attributes, and the committed ones is necessary. As we have shown in our work, we see great potential for BBS extensions when following the commit-and-prove paradigm. Include committed disclosure as part of the standard, would enable this component to be reused in a modular way. For instance, adding pseudonym support [KB25a] would not have to contain any BBS specifics anymore but rather be stated as an independent module.

Instantiating $\mathsf{NIZK}_{\mathsf{CD}}$ is possible using current draft standards. Issuing a zero-knowledge proof on linear relations in the group can be expressed by formulating a relation and prove/challenge/verification routines according to [Orr25b]. Making this proof non-interactive is achieved using the Fiat-Shamir transform, as described in [Orr25a].

Lastly, the PoP device binding proof requires numeric attributes, as the *dpk* is used for computations. The BBS core standard allows a custom attribute map [LKWL25] in Sec.4.1.2.1, which is sufficient to make PoP or range proofs over committed attributes compatible with the standard, when using our Pedersen-based approach.

C BBS-ECDSA: NIZKs for Bridging

Here we present the details of how we can prove knowledge of an ECDSA-based PoP for a public key attested through a BBS credential. We start by showing how to prove equality of committed values across groups, which allows us to move the committed public key from the BLS curve into a Tom curve. Then we show how the NIZK of knowing an ECDSA signature can be realized.

C.1 NIZK for Relation EQ

We next present the protocol for relation EQ of [OKMZ25]. The protocol is presented for fixed groups, namely $\mathsf{P\text{-}256}$ and $\mathsf{BLS12\text{-}381}$, and we instantiate the commitment in each with Pedersen. Recall that the protocol is parametrized; we use the parameters in the implementation from Dock Labs [Har25]. Recall the relation:

$$\mathsf{EQ}\left\{ (x, \rho, \rho') : \text{if } x < 2^{64} \text{ then } \begin{array}{l} C = \mathsf{Ped}_q.\mathsf{Commit}(pp_q, x; \rho) \\ C' = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, x; \rho') \end{array} \right\}$$

We denote with G, H the generators of $\mathsf{BLS12\text{-}381}$ used for Ped_q and with G', H' the generators of $\mathsf{BLS12\text{-}381}$ used for Ped_t. We assume all these parameters

$\mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{P}(pp_{\mathsf{EQ}}, x, w)$

Parse $pp_{\mathsf{EQ}} \leftarrow (\mathsf{BLS12\text{-}381}, G, H, \mathsf{P\text{-}256}, G', H', \mathsf{H}_{\mathsf{EQ}})$, $x \leftarrow (C, C')$ and $w \leftarrow (x, \rho, \rho')$
$k \xleftarrow{\$} \{0, \dots 2^{252} - 1\}$
$\tau \xleftarrow{\$} \mathbb{Z}_q$, $\tau' \xleftarrow{\$} \mathbb{Z}_t$
$K \leftarrow kG + \tau H$, $K' \leftarrow kG' + \tau' H'$
$c \leftarrow \mathsf{H}_{\mathsf{EQ}}(C, C', K, K')$ and if $c \notin \{0, \dots, 2^{128} - 1\}$ abort
$z \leftarrow k + cx$ and if $z \notin \{2^{244}, \dots, 2^{252} - 1\}$ abort
$s \leftarrow t + c\rho$, $s' \leftarrow t' + c\rho'$
Output $\pi \leftarrow (c, z, s, s')$

$\mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{V}(pp_{\mathsf{EQ}}, x, \pi)$

Parse $pp_{\mathsf{EQ}} \leftarrow (\mathsf{BLS12\text{-}381}, G, H, \mathsf{P\text{-}256}, G', H', \mathsf{H}_{\mathsf{EQ}})$, $x \leftarrow (C, C')$ and $\pi \leftarrow (c, z, s, s')$
$K \leftarrow zG + sH - cC$, $K' \leftarrow zG' + s'H' - cC'$,
Output $c \leftarrow \mathsf{H}_{\mathsf{EQ}}(C, C', K, K') \wedge z \in \{2^{244}, \dots, 2^{252} - 1\}$

Fig. 7: NIZK for relation EQ over Pedersen commitment scheme in different groups.

appear on the pp_{EQ}. We additionally use a hash function $\mathsf{H}_{\mathsf{EQ}} : \{0,1\}^* \to \{0, \dots, 2^{180} - 1\}$ which is also in the parameters. We can rewrite the relations as

$$\mathsf{EQ}\left\{(x, \rho, \rho') : \text{if } x < 2^{64} \text{ then } \begin{array}{l} C = xG + \rho H \\ C' = xG' + \rho' H' \end{array}\right\}$$

We first briefly describe how the proof works for each limb. The prover and verifier run a modified version of a standard Sigma protocol to prove that the representations of C_i, C_i' w.r.t. the corresponding commitment keys are equal (as integers).

The core insight of [COPZ22,OKMZ25] is that if we bound appropriately the values involved in the protocol, no $\mod q$ or $\mod t$ "wraparound" happens, and therefore the proof holds over the integers, which is enough to guarantee that the committed values are equal. We describe the protocol in Fig. 7.

C.2 NIZK for Proving Knowledge of an ECDSA Signature

Recall that an ECDSA signature σ on a message m under a public key $dpk = Q \in \mathsf{P\text{-}256}$ consists of elements $(\sigma_1, \sigma_2) \in \mathbb{Z}_p^2$. Let H be a hash function mapping messages to $\mathbb{Z}_p$. To verify the signature, the verifier computes $u_1 \leftarrow \mathsf{H}(m)\sigma_2^{-1}$ and $u_2 \leftarrow \sigma_1\sigma_2^{-1}$, computes $K \leftarrow u_1 G + u_2 Q$ and asserts that the x-coordinate of K equals σ_1 (in $\mathbb{Z}_p$). Following [FLM21] we present an alternative but equivalent

verification algorithm for ECDSA. The signature on m now consists of $(K \in \mathsf{P\text{-}256}, z = \sigma_2\sigma_1^{-1} \in \mathbb{Z}_p)$. The verifier now checks that $zK \stackrel{?}{=} \mathsf{H}(m)\sigma_1^{-1}G + Q$ where σ_1 is the x-coordinate of K in $\mathbb{Z}_p$. For completeness, we recall ECDSA signature scheme and the alternative verification in Appendix A.3.

In the device binding case, the signed message is some unique, public nonce sent by the verifier. This allows us revealing parts of the signature; concretely, we can give the value K on the clear without compromising privacy. Indeed, for any public key $\tilde{Q}$, there exists a value $\tilde{z}$ such that $\tilde{z}K = \mathsf{H}(m)\sigma_1^{-1}G + \tilde{Q}$; namely $\tilde{z} = \log_K(\mathsf{H}(m)\sigma_1^{-1}G + \tilde{Q})$. The message nonce is also a fresh random value, so it carries no information about the public key.

We can now reduce the problem of signature verification to two simple zero knowledge proofs. Let Q be the public key of the user. The user gets the signature (K, z) from the SE, computes $Z = zK$ and does the following:

- She sends K and the following commitments (we omit the randomness) to the verifier[10]:

$$C_{Qx} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_{\mathsf{Com}_t}, Q.x), \quad C_{Qy} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_{\mathsf{Com}_t}, Q.y)$$
$$C_{Zx} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_{\mathsf{Com}_t}, Z.x), \quad C_{Zy} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_{\mathsf{Com}_t}, Z.y)$$

$$C_z \leftarrow \mathsf{Ped}_p.\mathsf{Commit}(pp_{\mathsf{Com}_p}, z),$$

- She produces a Scalar Multiplication proof π_{SM} for the statement $Z = zK$.
- She produces a Point Addition Proof π_{PA} for the statement $Z = H(m)\sigma_1^{-1}G + Q$.

Note that the two proofs π_{PA} and π_{SM} constitute a proof of knowledge of an ECDSA signature on nonce under the public key committed in C_{dpk}. We formally present the relations next and give an overview of how to efficiently implement the corresponding NIZKs based on [FLM21,CLR24].

$$\mathsf{PA}\left\{\left(\left\{x_i, y_i, r_i, s_i\right\}_{i=1}^{3}\right) : \begin{array}{c} \left.\begin{array}{l} C_{x,i} = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, x_i; r_i) \\ C_{y,i} = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, y_i; s_i) \\ P_i = (x_i, y_i) \in \mathsf{P\text{-}256} \end{array}\right\}_{i=1}^{3} \\ P_1 + P_2 = P_3 \end{array}\right\}$$

$$\mathsf{SM}\left\{(\omega, r_\omega, x, r, y, s) : \begin{array}{c} C_x = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, x; r) \\ C_y = \mathsf{Ped}_t.\mathsf{Commit}(pp_t, y; s) \\ C_\omega = \mathsf{Ped}_p.\mathsf{Commit}(pp_p, \omega; r_\omega) \\ \omega G = (x, y) \end{array}\right\}$$

We start with the point addition protocol. The point addition on P-256 is a purely $\mathbb{Z}_t$ computation and therefore *native on* T-256. In particular, $P_1 = P_2 + P_3$

[10] Recall that in our construction we assume the public key is committed in limbs. We omit this here for simplicity and assume we have commitments to the coordinates $Q.x, Q.y$. The latter can be derived by homomorphically combining the commitments corresponding to the limbs.

can be expressed as a simple system of degree 2 equations which can be proven quite efficiently. Denoting with (x_i, y_i) the coordinates of P_i, the relation holds iff there exist $\lambda, \tau \in \mathbb{Z}_t$ such that[11]

$$\begin{aligned}(x_2 - x_1)\lambda &= y_2 - y_1 \\ \lambda^2 &= x_1 + x_2 + x_3 \\ \lambda(x_1 - x_3) &= y_1 + y_3 \\ (x_2 - x_1)\tau &= 1\end{aligned}$$

It is well known (see for example [WTs+18, App. A]) how to efficiently prove such constraints on committed values in zero knowledge. [FLM21] and [CLR24] propose using simple sigma protocols for such proofs.

For proving scalar multiplication the situation is a bit different since it inherently involves a non-native value, the scalar $\omega \in \mathbb{Z}_p$ such that $\omega P = (x, y)$. [CLR24] uses yet again a sigma protocol but involving non-native operation. The idea goes as follows: the prover wants to prove that $\omega G = (x, y)$ for some committed values $\omega \in \mathbb{Z}_p$ and $x, y \in \mathbb{Z}_t$ respectively. It interacts with the verifier as follows:

- The prover first samples $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes $x', y' \xleftarrow{\$} \mathbb{Z}_t$ s.t. $\alpha G = (x', y')$ and the coefficients (x'', y'') of $(\alpha - \omega)G$
- The verifier sends as a challenge a random bit b.
- If $b = 0$ the prover opens α, x', y' and the verifier asserts the relation $\alpha G \stackrel{?}{=} (x', y')$ and if $b = 1$ the prover opens $(\alpha - \omega)$[12], x'', y'' and the verifier checks $(\alpha - \omega)G \stackrel{?}{=} (x'', y'')$.
- In parallel to the above the prover and verifier run the sigma protocol for Point Addition to assert that $(x', y') = (x'', y'') + (x, y)$.

Note that neither of the commitments to αG or $(\alpha - \omega)G$ alone reveal anything about ωG since they are randomly sampled statements. Furthermore, if $\omega G \neq (x, y)$ then either $\alpha G \neq (x', y')$ or $(\alpha - \omega)G \neq (x'', y'')$; therefore, a cheating prover will succeed with probability at most $1/2$. The point addition protocol that is run in parallel is also modified to have a bit challenge for efficiency reasons, as suggested in [CLR24]. Using standard techniques, in particular parallel repetition, the soundness error is reduced to the desired value. The final protocol is the Fiat-Shamir transform [FS87] of the above non-interactive protocol. We refer the reader to [CLR24] for details on how the NIZKs are instantiated.

Putting it all together. Having all the tools in our disposal we can implement a NIZK for the relation PoP by properly combining them. We first recall how to encode the ECDSA key *dpk* as BBS attributes. The key is a point in P-256 which

[11] We implicitly require that two of the points are on the curve. This is always the case for our application since the issuer verifies the correctness of the ECDSA public key.

[12] This commitment can be constructed via the homomorphic properties of Pedersen commitment and the commitments to α and ω.

means it consists of a pair $(x, y) \in \mathbb{Z}_t$. The user decomposes the key in limbs of 64 bits, namely, it computes

$$x = \sum_{i=0}^{3} x_i 2^{64 \cdot i}, \qquad y = \sum_{i=0}^{3} y_i 2^{64 \cdot i}, \qquad x_i, y_i \in \{0, 2^{64} - 1\}$$

and encodes each limb as an attribute of the BBS signature. We therefore need 8 dedicated attributes for the device public key dpk. We assume that the correctness of the limbs (i.e. that they are in the correct range, and they constitute a valid point of P-256) is asserted during issuance. In what follows we denote the limbs as $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$ and $\boldsymbol{y} = (y_0, y_1, y_2, y_3)$.

The user makes the BBS presentation and adds to the set of the committed attributes $\mathsf{S}_{\mathsf{Com}}$ the 8 dedicated attributes of the device public key. Denote the latter $\boldsymbol{C}_x, \boldsymbol{C}_y$. Then, the proof of possession starts to prove knowledge of a signature on a random nonce. The user computes the proof as follows:

- She commits (in limbs) to the dpk coordinates over T-256. Denote these commitments $\boldsymbol{C}'_x$ and $\boldsymbol{C}'_y$. She also creates a proof of same opening π_{EQ} for all limb pairs $(C_{x,i}, C'_{x,i})$ and $(C_{y,i}, C'_{y,i})$.
- She creates the $\mathbb{Z}_t$ commitment of $dpk = (x, y)$ as a linear combination of $\boldsymbol{C}'_x, \boldsymbol{C}'_y$ using the homomorphic properties of Pedersen commitment. Denote the resulting pair (C_x, C'_y) as a single commitment C'_{dpk}.
- She queries the SE to get a signature (σ_1, σ_2) on nonce and convert it to (K, z). She commits to the scalar element z and gives the element K in the clear. Denote the commitments and the value K as the commitment to the signature C_σ.
- She creates π_{ECDSA} that asserts that the committed signature in C_σ is valid for nonce under the key committed in C'_{dpk}. This proof internally uses PoK of point addition and scalar multiplication.

We present the NIZK construction assuming black box NIZKs for relations ECDSA and EQ in Figure 8. Two implementations of this approach already exist, one from Ubique [Amr25] and one from Dock Labs [Har25].

$\mathsf{NIZK}_{\mathsf{PoP}}.\mathsf{P}(pp_{\mathsf{PoP}}, x, w)$

Parse $x \leftarrow (\mathsf{nonce}, \{c_{x,i}, c_{y,i}\}_{i=1}^4))$ and $w \leftarrow (dpk, \sigma = (K, z), \{x_i, r_i, y_i, s_i\}_{i=1}^4))$.

The following hold:

- $\forall i \in [4]$: $c_{x_i} = \mathsf{Ped}_q.\mathsf{Commit}(pp_q, x_i, r_i)$ and $0 \leqslant x_i < 2^{64}$,
- $\forall i \in [4]$: $c_{y_i} = \mathsf{Ped}_q.\mathsf{Commit}(pp_q, y_i, s_i)$ and $0 \leqslant y_i < 2^{64}$,
- $Q = (x, y) = \left(\sum_{i=0}^{3} x_i 2^{64 \cdot i}, \sum_{i=0}^{3} y_i 2^{64 \cdot i}\right) \in \mathsf{P\text{-}256}$,
- (K, z) is a valid ECDSA signature on nonce under Q.

For $i \in \{1, \ldots, 4\}$ sample $r'_i, s'_i \xleftarrow{\$} \mathbb{Z}_t$ and compute

$$c'_{x,i} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_t, x_i, r'_i), \qquad c'_{y,i} \leftarrow \mathsf{Ped}_t.\mathsf{Commit}(pp_t, y_i, s'_i)$$

$$\pi^{\mathsf{EQ}}_{x,i} \leftarrow \mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{P}(pp_{\mathsf{EQ}}, (C_{x,i}, C'_{x,i}), (x_i, r_i, r'_i))$$

$$\pi^{\mathsf{EQ}}_{y,i} \leftarrow \mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{P}(pp_{\mathsf{EQ}}, (C_{y,i}, C'_{y,i}), (y_i, s_i, s'_i))$$

Prepare the PoK of an ECDSA signature:

$$x_{\mathsf{ECDSA}} \leftarrow (\mathsf{nonce}, \{C'_{x,i}, C'_{y,i}\}_{i=1}^4), \quad w_{\mathsf{ECDSA}} \leftarrow (K, z, x, y, \{r'_i, s'_i\}_{i=1}^4)$$

$$\pi_{\mathsf{ECDSA}} \leftarrow \mathsf{NIZK}_{\mathsf{ECDSA}}.\mathsf{P}(pp_{\mathsf{ECDSA}}, x_{\mathsf{ECDSA}}, w_{\mathsf{ECDSA}})$$

Output $\pi \leftarrow (\{C'_{x,i}, C'_{y,i}, \pi^{\mathsf{EQ}}_{x,i}, \pi^{\mathsf{EQ}}_{y,i}\}_{i=1}^4, \pi_{\mathsf{ECDSA}})$.

$\mathsf{NIZK}_{\mathsf{PoP}}.\mathsf{V}(pp_{\mathsf{PoP}}, x, \pi)$

Parse $x \leftarrow (\mathsf{nonce}, \{C_{x,i}, C_{y,i}\}_{i=1}^4))$ and $\pi \leftarrow (\{C'_{x,i}, C'_{y,i}, \pi^{\mathsf{EQ}}_{x,i}, \pi^{\mathsf{EQ}}_{y,i}\}_{i=1}^4, \pi_{\mathsf{ECDSA}})$.

For $i \in \{1, \ldots, 4\}$ compute

$$b_{x,i} \leftarrow \mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{V}(pp_{\mathsf{EQ}}, (C_{x,i}, C'_{x,i}), \pi^{\mathsf{EQ}}_{x,i})$$

$$b_{y,i} \leftarrow \mathsf{NIZK}_{\mathsf{EQ}}.\mathsf{V}(pp_{\mathsf{EQ}}, (C_{y,i}, C_{y,i}), \pi^{\mathsf{EQ}}_{x,i})$$

$x_{\mathsf{ECDSA}} \leftarrow (\mathsf{nonce}, \{C'_{x,i}, C'_{y,i}\}_{i=1}^4)$
$b_{\mathsf{ECDSA}} \leftarrow \mathsf{NIZK}_{\mathsf{ECDSA}}.\mathsf{V}(pp_{\mathsf{ECDSA}}, x_{\mathsf{ECDSA}}), \pi_{\mathsf{ECDSA}})$
Output $b_{x,1} \wedge b_{x,1} \wedge \ldots \wedge b_{x,4} \wedge b_{x,4} \wedge b_{\mathsf{ECDSA}}$

Fig. 8: NIZK for relation PoP assuming black box NIZKs for the relations EQ and ECDSA.

D Example Application: Age Proof

We demonstrate the practicality of our modular system with regard to real-world use through the example of privacy-preserving age verification.

D.1 Age Verification

In an age verification system, a user wants to convince a verifier that she has surpassed some age threshold (such as being over 18 years old) based on a digital credential from an issuer. We describe how to construct such a system from our modular approach. There are many ways of approaching age verification. We based our example on a recent standard of the EU.

To prevent credential "cloning", where a user copies her credential and shares it with other users, we rely on device binding. The user can then only generate valid presentations by querying the device it was issued for. Instead of using explicit revocation, the credentials have a limited lifetime. The user has to prove for each presentation that the credential was valid, and was not expired, at the time of presentation.

D.2 Credential Structure for Age Verification

Recently, a specification for an age verification system [Com25] was proposed by the European Commission. The actual age to be proven is represented using a simple boolean attribute, which signifies whether a user fulfills the age threshold. We base our credentials roughly on this specification and define the following attributes, which are signed using MSig:

`header`	Issuer-defined context
`dpk`	The device public key
`nbf`	Not Before, a timestamp
`exp`	Expires At, a timestamp
`over18`	A boolean age attribute
`overNN`	A boolean age attribute

It is specified that the verifier is convinced using a zero-knowledge proof ("A Relying Party SHOULD implement the Zero-Knowledge Proof verification mechanism" [Com25]), which should prove the following statements (See also A.7 Zero-Knowledge Proofs in [Com25]):

1. The credential is signed by the issuer (called the Attestation Provider).
2. The credential contains an attribute signifying age over 18, and its value is true (i.e. `over18` $= true$).
3. The device can generate a signature on a verifier-supplied nonce, and this signature can be verified with the public key (`dpk`) included as an attribute in the credential.

4. The proof was generated within the credential's validity period (i.e. between `nbf` and `exp`).
5. The verifier (called the Relying Party) can verify this proof (or proofs).

The additional `header` value is not explicitly mentioned in these requirements, we use it to enforce a credential schema (the order and data type of the attributes above) and scope (intended redemption contexts) per credential. Other than `over18`, other ages can be encoded to `overNN`, such as 16 or 65 [Com25]. Assuming a single `overNN` attribute, `header`, `dpk`, `nbf`, `exp`, `over18`, the count of attributes is thus at least six. This does not take into account the encoding, as a single credential attribute may be mapped to multiple attributes in a concrete scheme. We address the required encoding after presenting the modules.

D.3 Module Selection for Age Verification

To instantiate age verification using our approach, one needs the core committed disclosure module (CD), as well as the proof-of-possession module PoP for device binding and the range proof module RP for proving that the credential is valid and not expired.

Recall the committed disclosure relation, which can be used for claims (1) and (2), where $\boldsymbol{a} = (\texttt{header}, \texttt{dpk}, \texttt{nbf}, \texttt{exp}, \texttt{over18}, \texttt{overNN})$, $\mathsf{S}_{\mathsf{Rev}}$ contains the indices of `header`, `over18` and $\mathsf{S}_{\mathsf{Com}}$ contains the indices of `dpk`, `nbf`, `exp` :

$$\mathsf{CD}\left\{\left(\begin{array}{c}\boldsymbol{a}\in\mathcal{M}^{\ell}, cred,\\ \{\rho_i\}_{i\in\mathsf{S}_{\mathsf{Com}}}\end{array}\right) : \begin{array}{l}\mathsf{MSig.Vf}(ipk, \boldsymbol{a}, cred) = 1\\ \forall i\in\mathsf{S}_{\mathsf{Rev}} : a_{\mathsf{Rev},i} = \boldsymbol{a}[i]\\ \forall i\in\mathsf{S}_{\mathsf{Com}} : C_i = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \boldsymbol{a}[i]; \rho_i)\end{array}\right\}$$

Here, ipk is the issuer's (Attestation Provider's) public key, $cred$ is the credential. The committed attributes `dpk`, `nbf`, `exp` can now be reused for the next claims. We use the device binding module PoP for claim (3). Note that we make an encoding of `dpk` to a vector of integers explicit now.

$$\mathsf{PoP}\left\{\left(\{\texttt{dpk}_i, \rho_i\}_{i\in[t]}, \sigma\right) : \begin{array}{l}\mathsf{Sig.Vf}(dpk, \mathsf{nonce}, \sigma) = 1\\ \mathsf{Encode}(dpk) = \texttt{dpk}\\ \forall i\in[t]\ C_{\texttt{dpk}_i} = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \texttt{dpk}_i; \rho_i)\end{array}\right\}$$

For claim (4), the user has to prove that the credential has become valid in the past, and that its expiration is in the future. Let `nbf` be the Not Before attribute, representing the start of the validity period in a numeric format[13], `exp` be the expiry time and `time` be the current time, all in the same numeric format. We can use the range proof module RP for this task. Since the lower/upper bounds for the current time `time` are secret, we split the proof into two parts, where the bounds for each individual proof ($0, \texttt{time}$ and $\texttt{time}, 2^{64}$) are public. The maximum bound 2^{64} is a constant, independent of current time, which serves to minimize tracking potential, compared to $\texttt{time} + x$ approaches, where the offset

[13] Such as 64-bit Unix time, which we propose for the concrete instantiation.

x may be chosen by the verifier (Relying Party). Now, the same range proof system can be used to make two range proofs for the following two statements:

$$\mathsf{RP}\left\{(\mathtt{nbf}, \rho_{nbf}) : \begin{array}{l} 0 \leqslant \mathtt{nbf} < \mathtt{time}, \\ C_{\mathtt{nbf}} = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \mathtt{nbf}; \rho_{nbf}) \end{array}\right\}$$

$$\mathsf{RP}\left\{(\mathtt{exp}, \rho_{exp}) : \begin{array}{l} \mathtt{time} \leqslant \mathtt{exp} < 2^{64}, \\ C_{\mathtt{exp}} = \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \mathtt{exp}; \rho_{exp}) \end{array}\right\}$$

In conjunction, the two proofs effectively prove the following statement, which corresponds to claim (4):

$$(0 \leqslant)\ \mathtt{nbf} < \mathtt{time} \leqslant \mathtt{exp}\ (< 2^{64})$$

An example Present algorithm for the full system for proving `over18` is shown in Figure 9. The corresponding Verify algorithm can be constructed in the natural way, by first recomputing ctx, then verifying all the proofs $\pi_{\mathcal{R}}$, and finally checking whether $\mathtt{over18} \stackrel{?}{=} true$. A presentation for another age `overNN` is analogous, `overNN` and `over18` are then simply swapped in the procedure of Figure 9.

D.4 Concrete Instantiation

Having defined the modules and proof statements, we can now specify the concrete instantiation of the primitives. As presented in Section 4, BBS signatures and ECDSA can be used as MSig and Sig, respectively. BBS is parametrized by the pairing group, such as BLS12-381 (alternatively, BLS24-509 for a larger security margin [AFG24]). ECDSA for secure elements is performed over the NIST curve P-256, support for this curve is also prescribed by [Com25]. The proofs for CD and PoP are performed as described in the main document. The last primitive left to instantiate is the range proof, which can be accomplished by using Bulletproofs [BCC+16,BBB+18]. The concrete specification of integrating Bulletproofs into the system is left as future work.

The core BBS multi-message signature scheme can be implemented in accordance with [LKWL25] over BLS12-381. The standard requires the public strings `api_id`, `ciphersuite_id`, and `header` to be defined. For BLS12-381, the curve from the standard [LKWL25] suffices, leading to the following ciphersuite identification:

$$\mathtt{ciphersuite_id}||\text{"}\mathtt{BLS12381G1_XOF:SHAKE-256_SSWU_RO_}\text{"}$$

`api_id` defines the group and interfaces, which are mapping algorithms into the group. In our case, we propose `api_id` to be:

$$\mathtt{api_id} = \mathtt{ciphersuite_id}||\text{"}\mathtt{H2G_HM2S_S2S_}\text{"}$$

"`H2G_HM2S_`" refers to the specific hash-to-group and hash-message-to-scalar algorithms from the standard [LKWL25] and this string is part of the default

$\mathsf{Present}(ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \mathsf{S}_{\mathcal{R}}, \boldsymbol{a}, cred, x, w, ctx_V)$

Compute commitments

$$\forall i \in [t] \quad C_{\mathtt{dpk}_i} \leftarrow \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \mathtt{dpk}_i; \rho_{\mathtt{dpk}_i})$$

$$C_{\mathtt{nbf}} \leftarrow \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \mathtt{nbf}; \rho_{\mathtt{nbf}})$$
$$C_{\mathtt{exp}} \leftarrow \mathsf{Com.Commit}(pp_{\mathsf{Com}}, \mathtt{exp}; \rho_{\mathtt{exp}})$$

Prepare statements/witnesses for relations CD, PoP, RP (for `nbf` and `exp`):

$$x_{\mathsf{CD}} \leftarrow (ipk, \mathsf{S}_{\mathsf{Rev}}, \mathsf{S}_{\mathsf{Com}}, \{\mathtt{header}, \mathtt{over18}\}, \{\{C_{\mathtt{dpk}_i}\}_{i\in[t]}, C_{\mathtt{nbf}}, C_{\mathtt{exp}}\})$$

$$w_{\mathsf{CD}} \leftarrow (\mathtt{dpk}, \mathtt{nbf}, \mathtt{exp}, \mathtt{overNN}, cred, \{\{\rho_{\mathtt{dpk}_i}\}_{i\in[t]}, \rho_{\mathtt{nbf}}, \rho_{\mathtt{exp}}\})$$
$$x_{\mathsf{PoP}} \leftarrow (\{C_{\mathtt{dpk}_i}\}_{i\in[t]}, \mathsf{nonce}), \quad w_{\mathsf{PoP}} \leftarrow (\{\mathtt{dpk}_i, \rho_{\mathtt{dpk}_i}\}_{i\in[t]}, \sigma)$$
$$x_{\mathtt{nbf}} \leftarrow (\{C_{\mathtt{nbf}}\}, 0, \mathtt{time}), \quad w_{\mathtt{nbf}} \leftarrow (\{\mathtt{nbf}, \rho_{\mathtt{nbf}}\})$$
$$x_{\mathtt{exp}} \leftarrow (\{C_{\mathtt{exp}}\}, \mathtt{time}, 2^{64}), \quad w_{\mathtt{exp}} \leftarrow (\{\mathtt{exp}, \rho_{\mathtt{exp}}\})$$

Compute $ctx \leftarrow ctx_V || C_{\mathtt{dpk}_1} || \ldots || C_{\mathtt{dpk}_t} || C_{\mathtt{nbf}} || C_{\mathtt{exp}}$
Output $\pi = (\{\{C_{\mathtt{dpk}_i}\}_{i\in[t]}, C_{\mathtt{nbf}}, C_{\mathtt{exp}}\}, \pi_{\mathsf{CD}}, \pi_{\mathsf{PoP}}, \pi_{\mathsf{RP}_{\mathtt{nbf}}}, \pi_{\mathsf{RP}_{\mathtt{exp}}})$ where

$$\pi_{\mathsf{CD}} \leftarrow \mathsf{NIZK}_{\mathsf{CD}}.\mathsf{P}(pp_{\mathsf{CD}}, x_{\mathsf{CD}}, w_{\mathsf{CD}})[ctx]$$

$$\pi_{\mathsf{PoP}} \leftarrow \mathsf{NIZK}_{\mathsf{PoP}}.\mathsf{P}(pp_{\mathsf{PoP}}, x_{\mathsf{PoP}}, w_{\mathsf{PoP}})[ctx]$$
$$\pi_{\mathsf{RP}_{\mathtt{nbf}}} \leftarrow \mathsf{NIZK}_{\mathsf{RP}_{\mathtt{nbf}}}.\mathsf{P}(pp_{\mathsf{RP}_{\mathtt{nbf}}}, x_{\mathtt{nbf}}, w_{\mathtt{nbf}})[ctx]$$
$$\pi_{\mathsf{RP}_{\mathtt{exp}}} \leftarrow \mathsf{NIZK}_{\mathsf{RP}_{\mathtt{exp}}}.\mathsf{P}(pp_{\mathsf{RP}_{\mathtt{exp}}}, x_{\mathtt{exp}}, w_{\mathtt{exp}})[ctx]$$

Fig. 9: The Present algorithm for age verification. The age proven here is 18.

`api_id` value. Since our system needs an additional scalar-to-scalar mapping for the range proofs, we propose to extend `api_id` by "`S2S_`" for our system.

The BBS signature according to the draft standard requires the issuer-prescribed `header` value, which is to be disclosed at every presentation, serving as a form of issuer context. This context is public and is to be checked by the verifier at each presentation verification for correctness. We opt to use the following header value to restrict the redemption of credentials only to age verification applications:

$$\mathtt{header} = \mathtt{AGE_PROOF_V1_}schema$$

schema represents a canonical encoding of the data types and attributes specified below.

Encoding Attributes. BBS requires attributes to be encoded into scalars in $\mathbb{Z}_q$. [LKWL25] provides a hash-to-scalar interface for this. We now discuss how the attributes are mapped. Note that in BBS, the attributes are associated with their

position, i.e. the corresponding generator H_i. Thus, the order of attributes in conjunction with the mapping algorithm to the scalar field defines the attribute schema.

Header The header value can be hashed using hash-to-scalar. It should be prefixed with a distinct value, such as `0x020202`, so that even an empty header is always different from any boolean values.

Boolean values To encode booleans, i.e. for `over18` and `overNN`, one can use the hash-to-scalar with predefined values, such as `0x00` for false and $0x01$ for true, with a prefix of `0x01` to distinguish these from the header.

The previous two attributes can be encoded to BBS scalars using the hash-to-scalar method defined in [LKWL25]. However, PoP and RP require numeric types for attributes, e.g. for proving that a number is within a certain range. For this, we ensure that numeric attributes are less than q, and map them using the identity function to $\mathbb{Z}_q$.

Time points `nbf` (Not Before), and `exp` (Expires At) are time stamps. A possible instantiation for time stamps is a 64-bit integer representing the amount of seconds passed since 1970-01-01T00:00:00Z in UTC at time of issuance, see RFC 3339 for details. This gives number range that can be efficiently implemented in proofs, has enough values to represent any useful time range, and is compatible with existing standards. To encode such values, which have to preserve their numeric structure, we use the identity function. Since q of BLS12-381 is much larger than 2^{64}, every time stamp is also a valid scalar.

Device public key As discussed in Section 4, due to technical limitations, the 2x256-bit `dpk` has to be encoded as a vector $\mathsf{Encode}(\texttt{dpk}) = \texttt{dpk}$; concretely, each coordinate of `dpk` is encoded into 4 64-bit values. The encoding of the public key `dpk` thus consists of 8 64-bit numbers, which are directly mapped to scalars in $\mathbb{Z}_q$.

Summary The proposed system for proving age over 18 and one additional age threshold, over NN, needs 13 attributes. The attributes are `header`, `dpk` (encoded as 8 elements), `nbf`, `exp`, `over18`, `overNN`. Two attributes are always revealed: `header` and either `over18`, or `overNN`. The rest are committed. The age not being proven (either `over18`, or `overNN`) remains hidden.