Formalizing Privacy of Anonymous Credentials: A Provably Secure Framework with Predicate Proofs

Yu Zhang[1][0009−0001−2873−6138], Zongbin Wang[2][0009−0007−4173−8318], and Jian Hou[3⋆]

[1] Beijing Pushtime Technology Co., Ltd.
zy168612@163.com
[2] Beijing Infosec Technologies Co., Ltd.
pony_wang@hotmail.com
[3] 66037th Unit
houjianxiansheng@126.com

Abstract. Anonymous credentials enable privacy-preserving authentication but existing systems either lack support for predicate proofs or are tied to specific signature schemes without a formal framework. We propose the first constructive framework for anonymous credentials with native predicate proof support. The framework provides definitions of entities, operations, adversary models, and security properties—unforgeability, unlinkability, and minimal disclosure. To demonstrate its feasibility, we instantiate the framework with BBS signatures, Pedersen commitments, and ring signature based proofs. The instantiation yields compact credentials and efficient zero-knowledge predicate proofs. We prove that the instantiation satisfies all security properties under standard assumptions (q-SDH, discrete logarithm, and the zero-knowledge of the ring signature based proof). A performance evaluation confirms that credential issuance, verification, and predicate prove/verify are practical on standard hardware, with compact credentials and proofs whose communication cost grows modestly with the number and encoding length of proved attributes, and favorable runtime and storage relative to a CL-based baseline. Our framework provides a modular foundation for designing and analyzing anonymous credential systems with fine-grained attribute disclosure.

Keywords: Anonymous Credentials · Predicate Proofs · BBS Signatures · Minimal Disclosure

1 Introduction

Anonymous credentials [1–4] are a fundamental primitive that enables users to obtain credentials and demonstrate possession, with the defining property that transactions by the same user cannot be linked. This *unlinkability* distinguishes

⋆ Corresponding author

them from traditional X.509 certificates [5–7], which force full credential disclosure and allow trivial linking of multiple presentations. Anonymous credentials have become a cornerstone of privacy-preserving authentication with widespread applications [8–12].

Most existing anonymous credential systems support *selective disclosure* [13], allowing a user to reveal exact values of a subset of attributes while hiding the rest. However, selective disclosure still forces revealing precise values (e.g., a birthdate) even when the verifier only needs to know that an attribute satisfies a condition (e.g., age ≥ 18). This contradicts the principle of *minimal disclosure* [14–18]. To overcome this, a credential system should support both selective disclosure and *predicate proofs*: proving possession of a credential and that its attributes satisfy a given condition without disclosing the actual values.

We focus on two widely-used classes of predicate proofs: range proofs (e.g., proving $v \geq v_{\min}$) and set-membership proofs (e.g., proving $v \in S$). These are specific instantiations of the more general notion of predicate proofs.

The pioneering IDEMIX [8] showed that predicate proofs are realizable, but its constructions are *ad hoc* and tightly coupled with specific signatures. Recent works have attempted to provide general frameworks. For instance, Bobolz et al. [19] proposed a unified model for anonymous signatures that can express predicate proofs, yet it remains at the model level and lacks a concrete, efficient instantiation. Deuber et al. [20] introduced functional credentials with policy hiding and delegated verification, but their construction relies on predicate encryption, incurring high computational overhead and interactive protocols.

Thus, despite these advances, no existing framework provides a practical, deployable solution for anonymous credentials with predicate proofs—a gap that we address in this work.

1.1 Related Work

The seminal idemix system [8] and its CL signatures [1, 2] pioneered anonymous credentials with predicate proofs. However, CL-based constructions are generally less efficient than pairing-based alternatives such as BBS$^+$.

BBS and BBS+ signatures [13, 21–27] offer constant-size signatures regardless of message count. Yet, none natively support predicate proofs; they only enable selective disclosure of exact values, forcing applications to reveal precise attributes even when only a condition is needed. This limitation motivates the need for a practical framework that combines BBS-style credentials with predicate proofs.

Recent works have attempted to address this gap at the framework level. Bobolz et al. [19] proposed Universal Anonymous Signatures (UAS), a unified framework for group signatures, anonymous credentials, and ring signatures that can encode predicate proofs via functional placeholders. However, UAS remains at the conceptual level and offers no concrete, efficient instantiation for predicate proofs; its generic NIZK-based construction incurs significant overhead and lacks performance evaluation.

Deuber et al. [20] introduced functional credentials, supporting arbitrary predicates with policy hiding and delegated verification via predicate encryption and an interactive proof-of-decryption protocol. However, its expressiveness is limited by the underlying predicate encryption: encoding boolean formulas incurs an exponential blowup per disjunctive clause. Moreover, the interactive protocol requires multiple rounds and verification takes seconds even for moderate parameters.

An alternative route to instantiate our framework is to employ circuit-based zero-knowledge proofs (e.g., zk-SNARKs, zk-STARKs) [28, 29], which offer stronger expressiveness by supporting arbitrary computations. We instead adopt ring signature based proofs, since the circuit construction and proof generation of such systems incur substantial overhead, even for the variants that avoid a trusted setup. Our approach provides a lightweight and trust-minimized solution for the common case of range and set-membership proofs.

1.2 Contributions

We make the following contributions.

- **A constructive framework for anonymous credentials with predicate proofs.** We provide formal definitions of entities, operations, adversary models, and security properties (unforgeability, unlinkability, and minimal disclosure), enabling modular security reductions.
- **A concrete instantiation with formal security proofs.** We instantiate the framework using BBS signatures, Pedersen commitments, and ring signature based proofs. The construction supports both selective disclosure and efficient predicate proofs (e.g., range and set membership) while building on BBS signatures for issuance and unlinkable presentations. We prove that the instantiation satisfies all security properties under standard assumptions.
- **Practical deployability.** We evaluate the computational and communication costs of our scheme through theoretical analysis and experiments. Credential issuance, holder verification, and predicate prove/verify all complete in milliseconds on standard hardware; compared with CL-based Idemix, predicate presentation is roughly an order of magnitude faster and credentials and proofs are substantially more compact.

2 Problem Statement and System Model

2.1 Problem Statement

We consider the problem of designing an anonymous credential system that is unlinkable and supports both selective disclosure and predicate proofs. The goal is to enable a user to prove possession of a credential while simultaneously proving that certain attributes satisfy a given predicate, without disclosing the underlying attribute values. We formalize this problem under the following adversarial and privacy constraints.

Adversarial Model. All participants except the issuer may be malicious and may deviate arbitrarily from the protocol, collude with others, or attempt to extract private information from proofs. The issuer is trusted only to issue credentials correctly and to keep attribute values confidential; beyond this, it is not trusted (e.g., it may refuse to issue or attempt to impersonate users). All communication channels are assumed to be confidential and authenticated (e.g., TLS), which is standard for privacy-preserving authentication protocols. Such authentication at the channel level is established using short-lived session identifiers rather than long-term user identities, and thus does not affect the protocol-level unlinkability guarantee.

Unlinkability. Multiple presentations of the same credential must be unlinkable. More precisely, no adversary should be able to determine whether a given proof was generated from one credential or another, when both satisfy the same predicate.

Privacy. A verifier learns nothing about the attribute values beyond the truth of the predicate θ being proved.

The central goal is to construct a protocol fulfilling these requirements with formal, provable privacy guarantees.

2.2 System Model

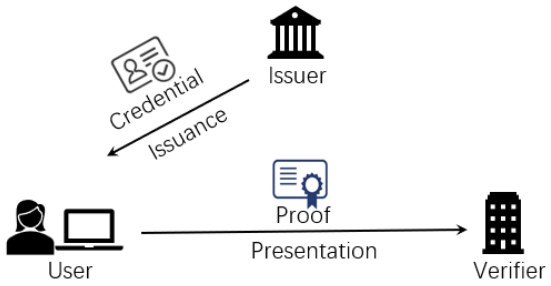

Fig. 1. System architecture

Entities and Workflow To address the problem defined above, we propose a constructive framework for anonymous credentials with predicate proofs. The system operates through three sequential phases:

- **System initialization phase:** Global public parameters are generated. Each **issuer(I)** independently creates a public/private key pair and publishes its public key.
- **Credential issuance phase:** A **user(U)** requests a credential from an issuer. The issuer validates the user's attribute values (e.g., via external evidence) and issues a credential that cryptographically binds the user's

master secret key to the attested attributes. The user stores the credential locally.

- **Credential presentation phase:** The user proves to a **verifier(V)** that she possesses a valid credential and that its attributes satisfy a predicate specified by the verifier, without revealing the actual attribute values. The verifier checks the proof and grants access only if verification succeeds.

Cryptographic Interface To protect the user's master secret key, we define algorithm Protect: on input pp and master secret key sk, it outputs a commitment Com and a zero-knowledge proof π_{open}:

$$\mathsf{Protect}(pp, sk) \to (Com, \pi_{\text{open}}).$$

Com hides sk, and π_{open} certifies correct formation.

The system's cryptographic workflow comprises six algorithms realizing three phases. Setup and KeyGen together form the system initialization phase, generating the global parameters and issuer keys, respectively. Credential issuance consists of the interactive protocols ApplyCred and GenCred. Credential presentation comprises the Prove and Verify algorithms.

We define our system as a framework specified by the abstract interfaces of its six algorithms (Setup, KeyGen, ApplyCred, GenCred, Prove, Verify) and the modular interactions among the entities. This abstraction separates the core security definitions from the concrete cryptographic instantiations, allowing different building blocks to be plugged in as long as they satisfy the required interface specifications.

We now define each algorithm formally.

- $\mathsf{Setup}(1^\lambda) \to pp$: Executed once by some party; takes security parameter λ and outputs global public parameters pp.
- $\mathsf{KeyGen}(pp) \to (pk_I, sk_I)$: Executed by an issuer; takes pp and outputs its public/private key pair; pk_I is made public.
- $\mathsf{ApplyCred}(pp, sk) \to (Com, \pi_{\text{open}})$: Executed by the user. It calls $\mathsf{Protect}(pp, sk)$ to obtain a commitment Com and a proof π_{open} for its master secret key sk, then outputs them.
- $\mathsf{GenCred}(pp, sk_I, \mathbf{v}, \mathsf{Proof}_v, Com, \pi_{\text{open}}) \to Cred$: Issuer verifies Proof_v (external evidence attesting to the correctness of the attribute vector $\mathbf{v}$, e.g., a digitally signed document) and π_{open} (proof for commitment Com). If both valid, outputs credential $Cred$ binding user's master secret key to $\mathbf{v}$; else $\perp$.
- $\mathsf{Prove}(pp, Cred, sk, \mathbf{v}, \theta, M) \to \pi_\theta$: here θ specifies both which attributes are proved, via the index set $M \subseteq \{2, \ldots, L\}$ (where L is the number of attribute slots fixed in pp), and the constraints on $\{v_i\}_{i \in M}$. If $\mathbf{v}$ does not satisfy θ, output $\perp$; otherwise output a zero-knowledge proof.
- $\mathsf{Verify}(pp, pk_I, \theta, \pi_\theta) \to \{0, 1\}$: Verifier checks π_θ and outputs 1 if valid, else 0.

2.3 Security Definitions

We now formalize the security properties of our framework. A scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ApplyCred}, \mathsf{GenCred}, \mathsf{Prove}, \mathsf{Verify})$ is secure if it satisfies completeness, unforgeability, unlinkability, and minimal disclosure. We define each property below.

Completeness. Completeness ensures honest participants can obtain credentials and generate valid proofs. We define two properties.

Definition 1 (Credential Issuance Completeness). *Π satisfies it if for any λ, $\mathrm{Adv}_{\Pi}^{\mathrm{ISS-COMP}}(\lambda) = \Pr[\mathrm{ISS-COMP}(\Pi, \lambda) = 1]$ is negligible. Experiment $\mathrm{ISS-COMP}(\Pi, \lambda)$:*

1. *The challenger $\mathcal{C}$ runs Setup and KeyGen to get pp, pk_I, sk_I.*
2. *$\mathcal{C}$ picks $sk, \mathbf{v}$, runs ApplyCred to get $(Com, \pi_{\mathrm{open}})$.*
3. *$\mathcal{C}$ runs GenCred to get Cred.*
4. *Output 1 if Cred $=\perp$, else 0.*

Definition 2 (Predicate Proof Completeness). *Π satisfies it if for any λ, $\mathrm{Adv}_{\Pi}^{\mathrm{PROOF-COMP}}(\lambda) = \Pr[\mathrm{PROOF-COMP}(\Pi, \lambda) = 1]$ is negligible. Experiment $\mathrm{PROOF-COMP}(\Pi, \lambda)$:*

1. *The challenger $\mathcal{C}$ takes a valid Cred with $\mathbf{v}, sk$, plus pp, pk_I.*
2. *$\mathcal{C}$ takes a predicate θ satisfied by $\mathbf{v}$.*
3. *Let M be the index set specified by θ.*
4. *$\mathcal{C}$ runs $\pi_\theta \leftarrow \mathsf{Prove}(pp, Cred, sk, \mathbf{v}, \theta, M)$.*
5. *Output 1 if $\mathsf{Verify}(pp, pk_I, \theta, \pi_\theta) = 0$, else 0.*

Oracle Definitions. To formalize the security properties, we introduce an oracle $\mathcal{O}^{\Pi}$ that abstracts the honest behavior of the system. It maintains the global state $(pp, pk_I, sk_I, \mathcal{R})$. The registry $\mathcal{R}$ stores entries of the form $(sk,\ \mathbf{v},\ Com,\ \pi_{\mathrm{open}},\ Cred)$, where fields are progressively filled during the issuance protocol. The oracle is instantiated with the system parameters and an initially empty registry $\mathcal{R}$. It is controlled by the challenger to enforce experiment restrictions and ensure all interactions follow the same honest protocol semantics. All queries from $\mathcal{A}$ are relayed through $\mathcal{C}$ to $\mathcal{O}^{\Pi}$, and the responses are returned via the same channel (i.e., through $\mathcal{C}$, which forwards them without modification). The oracle processes the following query types:

- $Q = (\mathsf{ApplyCred}, sk)$:
  - Compute $(Com, \pi_{\mathrm{open}}) \leftarrow \mathsf{ApplyCred}(pp, sk)$.
  - Create a new entry in $\mathcal{R}$ with $(sk, \mathbf{v} = \perp, Com, \pi_{\mathrm{open}}, Cred = \perp)$.
  - Return $(Com, \pi_{\mathrm{open}})$.
- $Q = (\mathsf{GenCred}, \mathbf{v}, \mathsf{Proof}_v, Com, \pi_{\mathrm{open}})$:
  - Verify Proof_v and π_{open}; abort if either is invalid.
  - Compute $Cred \leftarrow \mathsf{GenCred}(pp, sk_I, \mathbf{v}, \mathsf{Proof}_v, Com, \pi_{\mathrm{open}})$.
  - Locate the entry in $\mathcal{R}$ by Com and update it with $(\mathbf{v}, Cred)$.

  - Return *Cred*.
- $Q = (\mathsf{Prove}, \mathit{Cred}, \theta)$:
  - Retrieve the entry $(sk, \mathbf{v})$ associated with *Cred* from $\mathcal{R}$; if not found, abort.
  - If $\mathbf{v}$ does not satisfy θ, abort.
  - Let M be the index set specified by θ.
  - Compute $\pi_\theta \leftarrow \mathsf{Prove}(pp, \mathit{Cred}, sk, \mathbf{v}, \theta, M)$.
  - Return π_θ.

Unforgeability. Unforgeability ensures that no malicious user can produce a valid predicate proof for a credential that was not honestly issued, or for an attribute vector that does not satisfy the claimed predicate.

Definition 3 (Unforgeability). *A scheme Π satisfies unforgeability if for any probabilistic polynomial-time adversary $\mathcal{A}$ and any security parameter λ, the following advantage is negligible in λ:*

$$\mathrm{Adv}^{\mathrm{UF}}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\mathrm{UF}(\Pi, \mathcal{A}, \lambda) = 1\right].$$

The experiment $\mathrm{UF}(\Pi, \mathcal{A}, \lambda)$ *proceeds as follows:*

1. ***Initialization.*** *The challenger $\mathcal{C}$ generates the system parameters and issuer keys, initializes the oracle $\mathcal{O}^{\Pi}$ as defined above, and sends pp and pk_I to $\mathcal{A}$.*
2. ***Query Phase.*** *$\mathcal{A}$ may adaptively issue queries to the oracle $\mathcal{O}^{\Pi}$ (i.e., $\mathsf{ApplyCred}$, $\mathsf{GenCred}$, and Prove queries).*
3. ***Challenge.*** *$\mathcal{A}$ outputs a predicate proof π_θ for a predicate θ.*
4. ***Output.*** *The experiment outputs 1 if $\mathsf{Verify}(pp, pk_I, \theta, \pi_\theta) = 1$ and either:*
   - *π_θ was not generated from a credential issued by $\mathcal{O}^{\Pi}$ (i.e., no corresponding $\mathsf{GenCred}$ query produced the underlying credential), or*
   - *the attribute vector bound to the underlying credential does not satisfy θ.*

   Otherwise, the experiment outputs 0.

Unlinkability. Unlinkability ensures that a malicious verifier cannot distinguish whether a proof was generated from one credential or another, given that both satisfy the same predicate. We formalize this through the following left-or-right indistinguishability experiment.

Definition 4 (Unlinkability). *A scheme Π satisfies unlinkability if for any probabilistic polynomial-time adversary $\mathcal{A}$ and any security parameter λ, the following advantage is negligible in λ:*

$$\mathrm{Adv}^{\mathrm{UL}}_{\Pi,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathrm{UL}(\Pi, \mathcal{A}, \lambda) = 1\right] - \frac{1}{2}\right|.$$

The experiment $\mathrm{UL}(\Pi, \mathcal{A}, \lambda)$ *proceeds as follows:*

1. ***Initialization.*** *The challenger $\mathcal{C}$ generates the system parameters and issuer keys, initializes the oracle $\mathcal{O}^{\Pi}$ as defined above, and sends pp and pk_I to $\mathcal{A}$.*

2. ***Query Phase 1.*** *$\mathcal{A}$ may adaptively issue queries to $\mathcal{O}^{\Pi}$ (i.e., $\mathsf{ApplyCred}$, $\mathsf{GenCred}$, and Prove queries). All queries are relayed through $\mathcal{C}$ to $\mathcal{O}^{\Pi}$, and the responses are returned to $\mathcal{A}$.*
3. ***Challenge Phase.*** *$\mathcal{A}$ selects two distinct credentials $Cred_0$ and $Cred_1$ from those previously issued by $\mathcal{O}^{\Pi}$ (i.e., recorded in $\mathcal{R}$), along with a predicate θ satisfied by both corresponding attribute vectors. The challenger $\mathcal{C}$ picks a random bit $b \xleftarrow{\$} \{0,1\}$, lets M be the index set specified by θ, and computes $\pi_\theta \leftarrow \mathsf{Prove}(pp, Cred_b, sk_b, \mathbf{v}_b, \theta, M)$, where sk_b and $\mathbf{v}_b$ are the master secret key and attribute vector bound to $Cred_b$. $\mathcal{C}$ sends π_θ to $\mathcal{A}$.*
4. ***Query Phase 2.*** *$\mathcal{A}$ continues to issue adaptive queries to $\mathcal{O}^{\Pi}$ as in Query Phase 1.*
5. ***Output Phase.*** *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The experiment outputs 1 if $b' = b$, and 0 otherwise.*

Minimal Disclosure. Minimal disclosure captures the requirement that a predicate proof reveals nothing about the underlying attribute values beyond the fact that they satisfy the predicate θ. Intuitively, for any credential and predicate, there must exist a simulator that can construct an alternative set of secret values—different from the real ones but equally satisfying the predicate—such that the proof generated from these alternative secrets is indistinguishable from the real proof. We formalize this through the following indistinguishability experiment.

Definition 5 (Minimal Disclosure). *A scheme Π satisfies minimal disclosure if for any PPT adversary $\mathcal{A}$ and security parameter λ, the following advantage is negligible:*

$$\mathrm{Adv}^{\mathrm{MD}}_{\Pi,\mathcal{A}}(\lambda) = \left| \Pr[\mathrm{MD}(\Pi, \mathcal{A}, \lambda) = 1] - \frac{1}{2} \right|.$$

The experiment $\mathrm{MD}(\Pi, \mathcal{A}, \lambda)$ proceeds as follows:

1. ***Initialization.*** *The challenger $\mathcal{C}$ generates $pp \leftarrow \mathrm{Setup}(1^\lambda)$ and $(pk_I, sk_I) \leftarrow \mathrm{KeyGen}(pp)$, initializes the oracle $\mathcal{O}^{\Pi}$ as defined above, and gives pp, pk_I to $\mathcal{A}$.*
2. ***Query Phase.*** *$\mathcal{A}$ may adaptively issue queries to $\mathcal{O}^{\Pi}$ (ApplyCred, GenCred, and Prove).*
3. ***Challenge.*** *$\mathcal{A}$ selects a credential Cred previously issued by $\mathcal{O}^{\Pi}$ and a predicate θ such that the associated attribute vector $\mathbf{v}$ satisfies θ. The challenger $\mathcal{C}$ retrieves the corresponding master secret key sk from $\mathcal{R}$, picks a random bit $b \leftarrow \{0,1\}$, and proceeds as follows:*

$$(sk^*, \mathbf{v}^*) \leftarrow \begin{cases} (sk, \mathbf{v}), & \text{if } b = 0, \\ \mathcal{S}(pp, pk_I, \mathrm{Cred}, \theta, sk, \mathbf{v}), & \text{if } b = 1, \end{cases}$$

where $\mathcal{S}$ is a PPT simulator that, given a trapdoor τ (specified in the security proof), outputs alternative secrets $(sk', \mathbf{v}')$ such that a valid proof can be generated for θ.

Let M be the index set specified by θ. The challenger then computes $\pi_\theta \leftarrow \text{Prove}(pp, \text{Cred}, sk^*, \mathbf{v}^*, \theta, M)$ and sends π_θ to $\mathcal{A}$.

4. ***Response.*** *$\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*
5. ***Output.*** *The experiment outputs 1 if $b' = b$, and 0 otherwise.*

3 From Model To Instantiation: a Concrete Cryptographic Scheme

3.1 Mathematical Background

Non-Interactive Schnorr Proof Let $\mathbb{G}$ be a cyclic group of prime order q with generator G. The non-interactive Schnorr proof [30, 31] (via Fiat-Shamir) proves knowledge of x for $Y = [x]G$: $\mathsf{Prove}(Y, G, x)$ samples $r \leftarrow \mathbb{Z}_q$, computes $R = [r]G$, $c = \mathcal{H}(R\|Y)$, $s = r + cx \bmod q$, and outputs $\pi = (R, s)$; $\mathsf{Verify}(Y, G, \pi)$ recomputes $c = \mathcal{H}(R\|Y)$ and checks $[s]G = R + [c]Y$. Under the Discrete Logarithm (DL) assumption in the random oracle model (ROM), it satisfies completeness, soundness, and zero-knowledge.

Ring Signatures A ring signature scheme [32–34] enables a signer to anonymously sign a message on behalf of a ring of public keys $R_{PK} = \{PK_i = [x_i]G\}_{i=1}^n$ in a cyclic group $\mathbb{G}$ of order q with generator G. The signer owns secret key x_s for $PK_s \in R_{PK}$. The scheme consists of $\mathsf{Sign}(x_s, R_{PK}, m) \to \sigma$ and $\mathsf{Verify}(R_{PK}, m, \sigma) \to \{0,1\}$. It must satisfy correctness (valid signatures verify), anonymity (no PPT adversary identifies the signer with probability greater than $1/n$), and unforgeability (no PPT adversary forges a signature without a valid secret key).

Ring Signature Based Zero-Knowledge Proof (RingProof) A ring signature based range proof (RingProof) [35] allows a prover to show a committed value $v \in [0, 2^n - 1]$ without revealing it. For Pedersen commitment $C = [v]G + [r]H$, $\mathsf{Prove}(C, r, v) \to \pi$ decomposes v into bits and uses $\mathsf{RingSig.Sign}$ per bit; $\mathsf{Verify}(C, \pi)$ checks the ring signatures via $\mathsf{RingSig.Verify}$ and a consistency equation. Under the DL assumption, RingProof satisfies completeness, soundness, and zero-knowledge.

BBS Signatures BBS signatures are a pairing-based scheme that signs multiple messages into a single constant-size signature. They support efficient zero-knowledge proofs for selective disclosure and are self-randomizable, enabling unlinkable presentations. BBS is a W3C standard for verifiable credentials [13].

Recall the scheme. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be prime-order groups with pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and extra generators $h_1, \ldots, h_L \in \mathbb{G}_1$. The signer picks $sk \leftarrow \mathbb{Z}_q^*$ and sets $pk = [sk]g_2$. To sign $\mathbf{m} = (m_1, \ldots, m_L)$, sample random e and compute

$$A = \frac{1}{sk + e}\big(g_1 + \sum_{i=1}^{L} [m_i]h_i\big), \qquad \sigma = (A, e).$$

Verification checks $e(A, pk + [e]g_2) = e(g_1 + \sum_i [m_i]h_i, g_2)$.

3.2 Anonymous Credential System from BBS Signatures

We now instantiate the anonymous credential framework using BBS signatures. The system follows the six algorithms defined in Section 2.2.

System Initialization The system is initialized by executing the following algorithms.

Setup. The setup algorithm $\mathsf{Setup}(1^\lambda) \to pp$ is executed by some party:

1. Generate a description of bilinear groups $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with security parameter λ.
2. Choose random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.
3. Choose random generators $h_1, \ldots, h_L \in \mathbb{G}_1$, where L is the credential capacity (Table 1): h_1 binds the user master secret and $h_2, \ldots, h_L$ bind the $L-1$ credential attributes.
4. Output the public parameters $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \ldots, h_L)$.

Key Generation. The key generation algorithm $\mathsf{KeyGen}(pp) \to (pk_I, sk_I)$ is executed by each issuer:

1. Sample a random secret key $sk_I \xleftarrow{\$} \mathbb{Z}_q^*$.
2. Compute the corresponding public key $pk_I = [sk_I]g_2 \in \mathbb{G}_2$.
3. Output (pk_I, sk_I). The public key pk_I is made available to all users and verifiers.

Credential Issuance To obtain a credential for a master secret key sk and an attribute vector $\mathbf{v} = (v_2, \ldots, v_L) \in \mathbb{Z}_q^{L-1}$, the user and issuer execute the following two-step protocol.

Step 1: User Request. The user executes $\mathsf{ApplyCred}(pp, sk)$:

1. Compute $P = [sk]h_1$.
2. Produce a Schnorr zero-knowledge proof: $\pi \leftarrow \mathsf{Schnorr.Prove}(P, h_1, sk)$.
3. Send $\mathbf{v}, \mathsf{Proof}_v, P, \pi$ to the issuer, where Proof_v is external evidence for the attribute vector $\mathbf{v}$.

Step 2: Issuer Response. The issuer executes $\mathsf{GenCred}(pp, sk_I, \mathbf{v}, \mathsf{Proof}_v, P, \pi)$:

1. Verify the Schnorr proof: $\mathsf{Schnorr.Verify}(P, h_1, \pi)$. If invalid, output $\perp$.
2. Verify the external evidence Proof_v for the attribute vector $\mathbf{v}$. If invalid, output $\perp$.
3. Sample a random $e \xleftarrow{\$} \mathbb{Z}_q^*$.
4. Compute $A = \frac{1}{sk_I + e} \cdot \left(g_1 + P + \sum_{i=2}^{L} [v_i]h_i\right)$.
5. Output the credential $Cred = (A, e)$ and return it to the user.

The user then verifies the credential by checking the BBS verification equation. If the check passes, the user stores $Cred$ along with sk and $\mathbf{v}$.

Credential Presentation We now describe the Prove algorithm for credential presentation, which supports minimal disclosure by proving that a subset of attributes satisfy a given predicate without revealing their exact values. The algorithm follows the BBS proof of knowledge protocol [13], incorporating signature randomization and zero-knowledge predicate proofs.

Algorithm $\mathsf{Prove}(pp, Cred, sk, \mathbf{v}, \theta, M)$.

Input: Public parameters pp, a credential $Cred = (A, e)$, the user's master secret key sk, the attribute vector $\mathbf{v} = (v_2, \ldots, v_L)$, a predicate θ, and a set $M \subseteq \{2, \ldots, L\}$ of attribute indices to be proved.

Output: A non-interactive zero-knowledge predicate proof π_θ, or $\perp$ if $\mathbf{v}$ does not satisfy θ.

The algorithm proceeds as follows:

1. If $\mathbf{v}$ does not satisfy θ, output $\perp$.
2. **Randomize the master secret.** Set $v_1 = sk$. For each $i \in M$, sample $v_{1,i} \xleftarrow{\$} \mathbb{Z}_q$ and compute:
$$v_M = \sum_{i \in M} v_{1,i} \bmod q, \qquad v_I = v_1 - v_M \bmod q.$$
3. **Compute randomized commitments.** For each $i \in M$, compute:
$$P_i = [v_{1,i}]h_1 + [v_i]h_i.$$
The hidden part is $P_{\text{hid}} = [v_I]h_1$ (used internally).
4. **Randomize the BBS signature.** Let $I = \{2, \ldots, L\} \setminus M$. Sample $r \xleftarrow{\$} \mathbb{Z}_q^*$ and compute:
$$\overline{A} = [r]A, \qquad \overline{B} = [r]\left(g_1 + \sum_{i \in M} P_i + P_{\text{hid}} + \sum_{i \in I}[v_i]h_i\right) - [e]\overline{A}.$$
5. **Sample blinding factors.** Sample $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ and $\delta_i \xleftarrow{\$} \mathbb{Z}_q$ for each $i \in I \cup \{1\}$. Compute:
$$U = [\alpha]\left(g_1 + \sum_{i \in M} P_i\right) + [\beta]\overline{A} + \sum_{i \in I \cup \{1\}} [\delta_i]h_i.$$
6. **Generate predicate proofs.** For each $i \in M$, generate a zero-knowledge proof that v_i satisfies θ:
   - If θ is $v_i \geq v_{\min}$: $\pi_\theta^{(i)} = \mathsf{RingProof.Prove}\big(P_i - [v_{\min}]h_i,\ v_{1,i},\ v_i - v_{\min}\big)$.
   - If θ is $v_i \leq v_{\max}$: $\pi_\theta^{(i)} = \mathsf{RingProof.Prove}\big([v_{\max}]h_i - P_i,\ -v_{1,i},\ v_{\max} - v_i\big)$.
   - If θ is $v_{\min} \leq v_i \leq v_{\max}$: prove both bounds as above.

7. **Compute challenge and responses.** Compute the challenge:

$$c = \mathcal{H}\big(\overline{A}, \overline{B}, U, \{P_i\}_{i\in M}, \{\pi_\theta^{(i)}\}_{i\in M}\big),$$

where $\mathcal{H}$ is a cryptographic hash function. Then compute:
$s = \alpha + r \cdot c \bmod q, \qquad t = \beta - e \cdot c \bmod q, \qquad u_1 = \delta_1 + r \cdot v_I \cdot c \bmod q,$
$u_i = \delta_i + r \cdot v_i \cdot c \bmod q \quad (i \in I).$
8. **Output the proof.** Output:

$$\pi_\theta = \big(\overline{A}, \overline{B}, U, \{P_i\}_{i\in M}, \{\pi_\theta^{(i)}\}_{i\in M}, c, s, t, \{u_i\}_{i\in I\cup\{1\}}\big).$$

Algorithm Verify$(pp, pk_I, \theta, \pi_\theta)$.

Input: Public parameters pp, issuer public key pk_I, predicate θ, and proof π_θ.
Output: 1 if the proof is valid, 0 otherwise.

The verifier proceeds as follows:

1. Parse $\pi_\theta = (\overline{A}, \overline{B}, U, \{P_i\}_{i\in M}, \{\pi_\theta^{(i)}\}_{i\in M}, c, s, t, \{u_i\}_{i\in I\cup\{1\}})$.
2. For each $i \in M$, verify the predicate proof (for a range predicate $v_i \geq v_{\min}$):

$$\mathsf{RingProof.Verify}\big(P_i - [v_{\min}]h_i,\ \pi_\theta^{(i)}\big) \stackrel{?}{=} 1.$$

For other predicates (e.g., $\leq$ or interval), the verification is analogous.
3. Recompute the challenge:

$$c' = \mathcal{H}\big(\overline{A}, \overline{B}, U, \{P_i\}_{i\in M}, \{\pi_\theta^{(i)}\}_{i\in M}\big).$$

Check that $c' = c$. If not, output 0.
4. Verify the BBS proof equations:

$$e(\overline{A}, pk_I) \stackrel{?}{=} e(\overline{B}, g_2),$$

$$U + [c]\overline{B} \stackrel{?}{=} [s]\left(g_1 + \sum_{i\in M} P_i\right) + [t]\overline{A} + \sum_{i\in I\cup\{1\}} [u_i]h_i.$$

5. If all checks pass, output 1; otherwise output 0.

The randomization of the BBS signature components ensures unlinkability, while the zero-knowledge predicate proofs $\{\pi_\theta^{(i)}\}_{i\in M}$ ensure minimal disclosure, revealing nothing about the actual attribute values beyond the satisfaction of θ.

Extension to set membership. Our construction naturally supports set membership predicates $v_i \in S$ where $S = \{w_1, \ldots, w_k\}$ and $i \in M$. For each attribute index $i \in M$, the user constructs an independent ring signature as follows. For each candidate value $w \in S$, compute a public key $PK_{i,w} = P_i - [w]h_i$. The user, who holds the actual attribute value $v_i = w^* \in S$ and the randomness $v_{1,i}$, knows the discrete logarithm of PK_{i,w^*} with respect to h_1 (since $PK_{i,w^*} = [v_{1,i}]h_1$). A ring signature over the set $\{PK_{i,w}\}_{w\in S}$ then proves that $v_i \in S$ without revealing which value; its security reduces to the anonymity of the ring signature scheme. This approach requires no modification to the underlying credential format.

4 Security Analysis

Theorem 1. *Our scheme satisfies completeness (Definitions 1 and 2), unforgeability (Definitions 3), unlinkability (Definitions 4), and minimal disclosure (Definitions 5) under the correctness and security of the underlying building blocks: BBS signatures, Pedersen commitments, and RingProof.*

Proof sketch. Completeness follows directly from the correctness of BBS signatures, Schnorr proofs, and RingProof. Unforgeability reduces to BBS unforgeability or RingProof soundness. Unlinkability follows from BBS randomizability and the joint hiding/zero-knowledge of Pedersen commitments and RingProof. Minimal disclosure follows from a simulator that, given a trapdoor, constructs alternative secrets satisfying the same predicate with identical public commitments; indistinguishability then follows as in unlinkability. Full proofs are given in Appendix A.

5 Performance Evaluation

We compare schemes theoretically (Table 1) and report prototype timings for credential issuance and presentation on standard hardware.

5.1 Theoretical Analysis

We compare our scheme with IDEMIX [8] (CL-based; selective disclosure and predicate proofs) and BBS$^+$ [13] (pairing-based selective disclosure, but no native predicate proofs). Table 1 summarizes the comparison; all three schemes also provide unlinkability.

Table 1. Theoretical comparison of anonymous credential schemes. L: total attribute slots (including the master secret where applicable); $|M|$: attribute indices specified in θ; b: range bit-length; ℓ: bit-length of one public attribute base. *Holder storage*: on-device credential blob including attribute scalars.

Property	**IDEMIX**	**BBS$^+$**	**Ours**
Holder storage	$O(L)$	$O(L)$	$O(L)$
Public parameter size	$O(L \cdot \ell)$	$O(L \cdot \ell)$	$O(L \cdot \ell)$
Range proof size	$O(\|M\|)$	—	$O(\|M\| \cdot b)$
Minimal disclosure	Yes	No	Yes
Underlying assumptions	Strong RSA	q-SDH	q-SDH, DL

Holder storage scales as $O(L)$ in all cases (e.g., our on-device blob $(A, e, \mathbf{v})$). IDEMIX and BBS$^+$ (and hence ours) publish one public base per attribute slot (R_i vs. $h_i \in \mathbb{G}_1$), so public parameters scale as $O(L \cdot \ell)$ in all cases. The schemes differ in the per-base encoding length ℓ, not in the asymptotic form: at $\lambda \approx 128$-bit security, CL-IDEMIX uses a 3072-bit modulus, so each base R_i has $\ell \approx 3072$

bits. BBS$^+$ stores one $\mathbb{G}_1$ point per slot, with ℓ the on-wire encoding length of that point ($\ell \approx 384$ bits under compressed BLS12-381 serialization, as in Section 5.2), yielding substantially more compact public parameters for large L.

Our instantiation inherits BBS$^+$'s compact issuer setup and selective disclosure, and adds predicate proofs of size $O(|M| \cdot b)$—achieving the same minimal-disclosure goal as IDEMIX but with elliptic-curve parameter efficiency. Relative to BBS$^+$ alone, the gain is native predicate proofs without revealing exact attribute values.

5.2 Experimental Evaluation

Platform and software. All experiments were run on a single Linux host (kernel 5.15, x86_64), with both prototypes compiled at `-O2`. Our scheme uses C and blst [36] on BLS12-381; the Idemix baseline follows the IBM Identity Mixer specification [37], implemented with Silvia [38] (C++, GMP, OpenSSL). Both prototypes are single-threaded at the application level; each timed operation runs sequentially on one CPU core (no explicit multithreading in either benchmark harness). All reported runtimes are in milliseconds and are means over ten runs.

Settings. We compare at $\lambda \approx 128$ (BLS12-381 vs. 3072-bit Strong RSA). Unless stated otherwise, both prototypes use $n_{\text{attr}} = 8$ credential attributes and issuer setup with $L = n_{\text{attr}} + 1$ slots (Table 1), where one slot is reserved for the user master secret and n_{attr} slots for the attributes. Range predicates use $b = 8$ and interval $[0, 255]$ in our scheme, and IBM PROVERANGE with $attr_i \geq 0$ in Idemix under $L_m = 8$, so both sides encode attributes in the same byte range. Many deployments need only such encodings—for example, access-control clearance levels or eligibility tiers issued as small integers rather than full-width field elements. All attributes remain hidden (no selective disclosure); predicate cost is measured for $|M| \in \{1, 3, 5, 7\}$ range clauses.

Issuer initialization (one-time). Before any credential is issued, the issuer publishes public parameters and generates a long-term signing key. Our prototype uses `bbs_setup` and `bbs_keygen`; at $n_{\text{attr}} = 8$ ($L = 9$) the combined time is about 1.2 ms. Idemix performs the same step in one call, `generate_keypair`, which builds the Strong-RSA public parameters and the issuer key pair; at the same attribute budget ($L = 9$) it averages about 10,800 ms over ten runs ($\approx$ 8–13 s across runs; 3072-bit safe-prime generation dominates).

Storage. For our scheme, byte counts use compressed BLS12-381 serialization (48 B per $\mathbb{G}_1$ point, 96 B per $\mathbb{G}_2$ point), as in the W3C BBS$^+$ credential format. We report sizes at $n_{\text{attr}} = 8$ with one range clause ($|M| = 1$). Issuer public parameters occupy 672 B (ours) vs. 4,608 B (Idemix, minimal big-integer encodings of the modulus, S, Z, and attribute bases). Holder credentials: 368 B (ours: A, e, and scalars $v_1, \ldots, v_L$; slot v_0 is unused) vs. 928 B. Predicate proofs or presentations: 2,976 B vs. 6,022 B; both grow roughly linearly in $|M|$ (about

7.7 KiB vs. 15.0 KiB at $|M| = 3$). At $b = 32$ with $|M| = 1$, our predicate proof is about 8.5 KiB, larger than Idemix's presentation under either $L_m = 8$ or $L_m = 32$ (about 6.0 KiB).

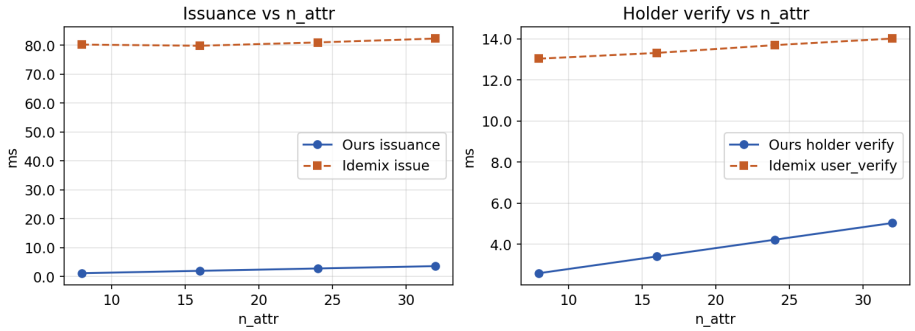

Fig. 2. Per-credential issuance and holder verification time (ms) vs. n_{attr}.

Credential issuance and holder verification. Figure 2 plots per-credential *issuance* and *holder verification* time (ms) versus $n_{\text{attr}} \in \{8, 16, 24, 32\}$. At $n_{\text{attr}} = 8$, our issuance and holder verification take about 1.2 ms and 2.6 ms, respectively; Idemix takes about 84 ms and 14 ms. Ours rises mildly (to about 3.6/5.0 ms at $n_{\text{attr}} = 32$); Idemix issuance stays ≈ 84 ms (modular exponentiation dominates) and holder verification ≈ 14 ms.

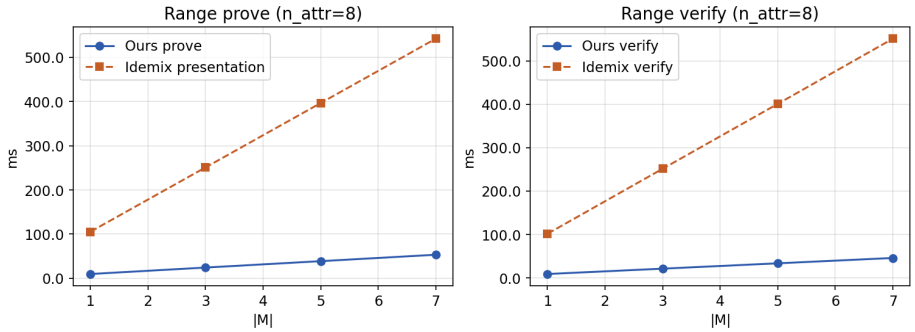

Fig. 3. Predicate prove and verify time (ms) vs. $|M|$ ($n_{\text{attr}} = 8$; $b = 8$, Idemix $L_m = 8$).

Predicate proving and verification. Figure 3 plots prove and verify time (ms) vs. $|M|$ at $n_{\text{attr}} = 8$ with $b = 8$ (ours) and $L_m = 8$ (Idemix). At $|M| = 1$,

ours averages about 9.9 ms (prove) and 8.9 ms (verify), versus about 105 ms and 102 ms for Idemix—roughly 10× faster on elliptic-curve arithmetic versus Strong-RSA modular work with different range-proof designs (Table 1). The advantage persists as $|M|$ grows (e.g., about 53 ms vs. 542 ms for proving at $|M| = 7$). As a sensitivity check at 32-bit encodings ($b = 32$, Idemix $L_m = 32$, still $|M| = 1$), our costs rise to about 26 ms and 22 ms while Idemix remains near 107 ms and 102 ms—still several times faster, with larger proofs on our side (about 8.5 KiB vs. about 6.1 KiB).

6 Conclusion

We presented the first constructive framework for anonymous credentials that natively supports predicate proofs, including definitions of entities, operations, adversary models, and security properties (unforgeability, unlinkability, minimal disclosure). To demonstrate feasibility, we instantiated the framework with BBS signatures, Pedersen commitments, and ring signature based proofs, yielding compact credentials and efficient zero-knowledge predicate proofs. We proved that the instantiation satisfies all security properties under standard assumptions. Experimental evaluation confirmed practicality: predicate costs scale with $|M|$ and the chosen range bit-length b; $b = 8$ suffices for common byte-sized policy encodings, and even $b = 32$ with Idemix $L_m = 32$ remains faster on prove/verify while Idemix stays more compact at large b.

Future work includes support for expressive predicates, post-quantum instantiations, and integration into decentralized identity systems and standardization efforts.

References

1. Camenisch J, Lysyanskaya A. A signature scheme with efficient protocols. In: *Proceedings of SCN 2002*. Springer, 2002, 268–289.
2. Camenisch J, Lysyanskaya A. Signature schemes and anonymous credentials from bilinear maps. In: *Proceedings of CRYPTO 2004*. Springer, 2004, 56–72.
3. Kakvi S A, Martin K M, Putman C, et al. Sok: anonymous credentials[C]//International Conference on Research in Security Standardisation. Cham: Springer Nature Switzerland, 2023: 129-151.
4. Baldimtsi F, Lysyanskaya A. Anonymous credentials light[C]//Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. 2013: 1087-1098.
5. Holz R, Braun L, Kammenhuber N, et al. The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements[C]//Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. 2011: 427-444.
6. Bastian P, Stöcker C, Schwalm S. Combination of x509 and DID/VC for inheritance properties of trust in digital identities[C]//Open Identity Summit 2022. Gesellschaft für Informatik eV, 2022: 137-142.

7. Larisch J, Aqeel W, Lum M, et al. Hammurabi: A framework for pluggable, logic-based X. 509 certificate validation policies[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022: 1857-1870.
8. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 21–30 (2002)
9. Camenisch, J., Groß, T.: Efficient attributes for anonymous credentials. In: Proceedings of the 15th ACM Conference on Computer and Communications Security,pp. 345–356 (2008)
10. Pussewalage, H.S.G., Oleshchuk, V.A.: An anonymous delegatable attribute-basedcredential scheme for a collaborative e-health environment. ACM Trans. InternetTechnol. (TOIT) 19(3), 1–22 (2019)
11. Förster, D., Kargl, F., Löhr, H.: PUCA: a pseudonym scheme with user-controlledanonymity for vehicular ad-hoc networks (VANET). In: 2014 IEEE Vehicular Net-working Conference (VNC), pp. 25–32. IEEE (2014)
12. Doesburg, J., Jacobs, B., Ringers, S.: Using IRMA for small scale digital elections(2020)
13. Tessaro S, Zhu C. Revisiting BBS signatures[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer Nature Switzerland, 2023: 691-721.
14. Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr). A practical guide, 1st ed., Cham: Springer International Publishing, 2017
15. Zaeem R N, Barber K S. The effect of the GDPR on privacy policies: Recent progress and future promise. ACM Transactions on Management Information Systems (TMIS), 2020, 12(1): 1-20.
16. Albrecht J P. How the GDPR will change the world. Eur. Data Prot. L. Rev., 2016, 2: 287.
17. ISO/IEC. (2022). Information security, cybersecurity and privacy protection—Information security management systems—Requirements (ISO/IEC 27001:2022). International Organization for Standardization.
18. National Institute of Standards and Technology (NIST). (2020). NIST privacy framework: A tool for improving privacy through enterprise risk management (Version 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.CSWP.01162020
19. Bobolz J, Diaz J, Kohlweiss M. Foundations of anonymous signatures: Formal definitions, simplified requirements, and a construction based on general assumptions[C]//International Conference on Financial Cryptography and Data Security. Cham: Springer Nature Switzerland, 2024: 121-139.
20. Deuber D, Maffei M, Malavolta G, et al. Functional credentials[J]. Proceedings on Privacy Enhancing Technologies, 2018.
21. Boneh D, Boyen X, Shacham H. Short group signatures[C]//Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 41-55.
22. Camenisch J, Lysyanskaya A. Signature schemes and anonymous credentials from bilinear maps[C]//Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 56-72.
23. Chairattana-Apirom R, Tessaro S. On the Concrete Security of BBS/BBS+ Signatures[C]//International Conference on the Theory and Application of Cryptology and Information Security. Singapore: Springer Nature Singapore, 2025: 402-434.
24. Flamini A, Lee E, Lysyanskaya A. Multi-holder anonymous credentials from bbs signatures[C]//Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2025: 325-357.

25. Doerner J, Kondi Y, Lee E, et al. Threshold BBS+ signatures for distributed anonymous credential issuance[C]//2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023: 773-789.
26. Desmoulins N, Dumanois A, Kane S, et al. Making BBS Anonymous Credentials eIDAS 2.0 Compliant[J]. Cryptology ePrint Archive, 2025.
27. Au M H, Susilo W, Mu Y. Constant-size dynamic k-TAA[C]//International conference on security and cryptography for networks. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006: 111-125.
28. Liang J, Hu D, Wu P, et al. SoK: Understanding zk-SNARKs: The Gap Between Research and Practice[C]//34th USENIX Security Symposium (USENIX Security 25). 2025: 2085-2104.
29. Oude Roelink B, El-Hajj M, Sarmah D. Systematic review: Comparing zk-SNARK, zk-STARK, and bulletproof protocols for privacy-preserving authentication[J]. Security and Privacy, 2024, 7(5): e401.
30. Hao, Feng. Schnorr non-interactive zero-knowledge proof. No. rfc8235. 2017.
31. Camenisch, J., Kiayias, A., Yung, M. (2009). On the Portability of Generalized Schnorr Proofs. In: Joux, A. (eds) Advances in Cryptology - EUROCRYPT 2009
32. Bender A, Katz J, Morselli R. Ring signatures: Stronger deffnitions, and constructions without random oracles//Theory of Cryptography Confer-ence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006: 60-79.
33. Rivest, R.L., Shamir, A., Tauman, Y. (2006). How to Leak a Secret: Theory and Applications of Ring Signatures. In: Goldreich, O., Rosen-berg, A.L., Selman, A.L. (eds) Theoretical Computer Science. Lecture Notes in Computer Science, vol 3895. Springer, Berlin, Heidelberg.
34. Wang L, Zhang G, Ma C. A survey of ring signature. Frontiers of Electrical and Electronic Engineering in China, 2008, 3(1): 10-19.
35. Noether S, Mackenzie A. Ring conffdential transactions. Ledger, 2016, 1: 1-18.
36. https://github.com/supranational/blst
37. Bichsel P, Binding C, Camenisch J, et al. Cryptographic protocols of the identity mixer library[M]//Technical Report. 2009, 99740: 3730.
38. https://github.com/credentials/silvia

A Full Security Proofs

This appendix provides the full proofs for Theorem 1 in Section 4.

A.1 Proof of Completeness

Completeness consists of two properties: credential issuance completeness and predicate proof completeness.

Credential Issuance Completeness. In the issuance protocol, the user computes $P = [sk]h_1$ and a Schnorr proof π proving knowledge of sk. The Schnorr proof is complete, so an honest prover always convinces the issuer. The issuer verifies π and the external evidence Proof_v for $v_2, \ldots, v_L$. Since both are valid, the issuer samples a random e and computes

$$A = \frac{1}{sk_I + e}\left(g_1 + P + \sum_{i=2}^{L}[v_i]h_i\right).$$

By the correctness of BBS signatures, the resulting $Cred = (A, e)$ satisfies the verification equation; thus $\mathrm{Adv}_{\Pi}^{\mathrm{ISS-COMP}}(\lambda)$ is negligible.

Predicate Proof Completeness. The Prove algorithm constructs $\overline{A}, \overline{B}, U$ and responses $s, t, u_1, \{u_i\}_{i \in I}$ exactly such that the verification equations

$$e(\overline{A}, pk_I) = e(\overline{B}, g_2), \qquad U + [c]\overline{B} = [s]\Big(g_1 + \sum_{i \in M} P_i\Big) + [t]\overline{A} + \sum_{i \in I \cup \{1\}} [u_i]h_i$$

hold by the bilinearity of the pairing and the definitions of the components. The ring signature based proofs $\pi_\theta^{(i)}$ are generated using RingProof.Prove, which is complete. Hence the verifier accepts the proof, and thus the incompleteness advantage is negligible.

A.2 Proof of Unforgeability

The adversary $\mathcal{A}$ outputs a valid proof π_θ. We distinguish two mutually exclusive and exhaustive cases, based on whether the credential underlying the forged proof was ever issued by the oracle.

Case 1: The underlying credential was never issued. In this case, the adversary outputs a valid proof without having obtained the corresponding credential via a GenCred query. By the special soundness of the BBS$^+$ proof-of-knowledge protocol in [13], from two accepting transcripts with distinct challenges we can extract the original BBS signature (A, e) on the attribute vector $\mathbf{v}$. Since no GenCred query ever issued a credential for $\mathbf{v}$, this (A, e) constitutes a forgery against the BBS signature scheme. A standard reduction yields an adversary $\mathcal{B}_1$ against BBS unforgeability, so the probability of this case is bounded by $\mathrm{Adv}_{\mathrm{BBS}}^{\mathrm{uf}}(\mathcal{B}_1, \lambda)$.

Case 2: The underlying credential was issued, but the claimed predicate is false for its attribute vector. Otherwise the credential was issued honestly but θ is false. Consider the case where θ is a range predicate $v_i \geq v_{\min}$ (other predicates are analogous). If $v_i < v_{\min}$, then the statement proved by the RingProof (i.e., that $P_i - [v_{\min}]h_i$ commits to a value in $[0, 2^m - 1]$) is false. The adversary nevertheless produces a valid proof $\pi_\theta^{(i)}$. This directly violates the soundness of the RingProof scheme, which guarantees that no PPT adversary can produce a valid proof for a false statement except with negligible probability. Therefore, the probability of this case is bounded by $\mathrm{Adv}_{\mathrm{RingProof}}^{\mathrm{sound}}(\mathcal{B}_2, \lambda)$.

Since the two cases are exhaustive, the total advantage of $\mathcal{A}$ is bounded by the sum of the two advantages, plus negligible terms. Hence the scheme is unforgeable.

A.3 Proof of Unlinkability

In the unlinkability experiment, $\mathcal{A}$ receives a proof π_θ generated from one of two credentials $\mathrm{Cred}_0, \mathrm{Cred}_1$. To distinguish which credential was used, $\mathcal{A}$ must distinguish either the BBS components or the predicate-related components.

For the BBS components $\overline{A}, \overline{B}, U, c, s, t, \{u_i\}_{i \in I \cup \{1\}}$, BBS randomizability ensures that their distributions from Cred_0 and Cred_1 are computationally indistinguishable, yielding the bound $\mathrm{Adv}^{\mathrm{rand}}_{\mathrm{BBS}}(\mathcal{B}_1, \lambda)$.

For predicate components, uniform blinding makes $\{P_i\}$ statistically indistinguishable; RingProof zero-knowledge yields $\mathrm{Adv}^{\mathrm{hide+zk}}_{\mathrm{Pred}}(\mathcal{B}_2, \lambda)$. Summing with the BBS term bounds $\mathrm{Adv}^{\mathrm{UL}}_{\Pi,\mathcal{A}}(\lambda)$ under q-SDH and DL; hence the scheme satisfies unlinkability.

A.4 Proof of Minimal Disclosure

We construct a simulator $\mathcal{S}$ that, given a trapdoor τ (the discrete logarithms $\log_{h_1} h_i$), outputs alternative secrets $(sk', \mathbf{v}')$ such that:

- $\theta(\mathbf{v}') = 1$, and
- $[sk']h_1 + \sum_{i=2}^{L} [v'_i]h_i = [sk]h_1 + \sum_{i=2}^{L} [v_i]h_i$.

Hence the same $\mathrm{Cred} = (A, e)$ verifies under both $(sk, \mathbf{v})$ and $(sk', \mathbf{v}')$.

In the minimal disclosure experiment (Definition 5), the challenger picks a random bit b. If $b = 0$, it uses the real secrets $(sk, \mathbf{v})$; if $b = 1$, it invokes $\mathcal{S}$ to obtain $(sk', \mathbf{v}')$. Let M be the index set specified by θ; in both cases the challenger computes π_θ via $\mathsf{Prove}(pp, \mathrm{Cred}, sk^*, \mathbf{v}^*, \theta, M)$.

Both branches satisfy θ; by the unlinkability argument (BBS randomization and RingProof zero-knowledge), the proofs are computationally indistinguishable, so $\mathrm{Adv}^{\mathrm{MD}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible (Pedersen hiding and RingProof ZK).