Reusable Anonymous Credentials with Flow-Mode Unlinkability

***Abstract*—Anonymous credentials let a holder prove certified attributes and verifier predicates without revealing the credential or linking sessions. SNARK-based credentials support expressive predicates and deployed signatures such as ECDSA and BLS, but remain stateless: each presentation re-proves the credential relation and predicates from scratch. Reusable presentations amortize this cost, but create a leakage channel outside standard unlinkability: a verifier may detect that two tokens came from the same evolving reuse chain.**

We introduce *reusable anonymous credentials*, where a holder maintains a compact private state and derives fresh presentation tokens by rerandomization or incremental predicate update. Our security model introduces *flow-mode unlinkability*, which requires tokens from the same reuse chain to be indistinguishable from independently generated tokens under the same public schedule. This captures colluding verifiers who pool transcripts across repeated presentations.

SHIFT realizes this model with two components. SUPERHALO is a function-hiding non-uniform incrementally verifiable computation (IVC) from Halo-style KZG accumulation. Standard multi-function IVC reveals the active function because only one accumulator slot changes; SUPERHALO instead updates all slots with dummy work and hides the update trace. SHIFT also uses proof composition for signature possession: a Σ-protocol proves expensive BLS pairing checks outside the recursive circuit, while the IVC binds the proof to the credential digest, predicates, and session context.

Our Rust implementation evaluates BLS credentials and an ECDSA variant on commodity hardware. Rerandomization takes about $0.65\,$s independent of attribute length and predicate size, proof size is below $2\,$KB, and verification is below $50\,$ms. Incremental predicate update reduces prover time by up to $3.5\times$ compared with the proof-of-possession baseline of Woo *et al.* (S&P 2025) in the evaluated setting.

1. Introduction

Identity management is a central component of secure digital systems, yet most deployed solutions rely on centralized identity providers and repeated disclosure of user data. Decentralized identity systems aim to shift control to users by representing identity claims as verifiable credentials and by using cryptographic mechanisms to minimize disclosure.

Anonymous credentials (ACs) are a key primitive in this setting. An AC system involves an issuer, a holder, and a verifier. The issuer attests to a holder's attributes by issuing a credential, and the holder later produces a presentation that convinces the verifier that the credential is valid and that specified predicates over the certified attributes hold (*e.g.*, $\mathrm{age} \geq 18$). The privacy objective is twofold: a presentation should reveal no information beyond what the proved predicates imply, and it should be unlinkable across sessions. These properties enable privacy-centric applications such as anonymous web browsing [1], privacy-preserving blacklisting [2], and combating misinformation [3].

Despite substantial interest, deploying ACs in practice remains challenging. Existing approaches force an unfavorable trade-off among predicate expressiveness, interoperability with widely deployed signatures, and the cost of generating presentations.

Previous solutions and limitations. Classical AC systems achieve efficient presentations by using signature schemes with algebraic structure, including BBS+ signatures [4], unique ring signatures [5], Pointcheval–Sanders signatures [6], and structure-preserving signatures on equivalence classes [7], as used in later AC constructions [8], [9], [10]. These schemes exploit signature rerandomization and tailored Σ-protocols [11] to obtain compact, efficient presentations. Their efficiency, however, is tied to algebraic predicate classes such as equality, selective disclosure, and range proofs; richer policies require additional machinery or direct disclosure. They also do not directly interoperate with widely deployed elliptic-curve digital signature algorithm (ECDSA) and Boneh–Lynn–Shacham (BLS) credentials.

A second line uses general-purpose zk-SNARKs to prove credential validity and predicate satisfaction for arbitrary NP relations [1], [12]. This supports expressive policies and legacy signatures, but the prover is the bottleneck: verifying a legacy signature inside a SNARK circuit costs about 2^{18}–2^{22} constraints for P-256 or BN254, leading to per-session proving times from seconds to tens of seconds. Recent proof-composition techniques reduce the cost of proving legacy-signature possession [13], [14], but the presentation model remains stateless: each session re-proves the credential relation and the relevant predicates from scratch.

Design point. SHIFT targets the simultaneous requirements of compatibility with deployed credential signatures, expressive predicate support, succinct verification, reusable presentations, and unlinkability across tokens derived from the same reuse flow. Simpler design points are possible if one requirement is removed: native anonymous-credential signatures give cleaner algebraic presentations but do not directly handle legacy certificates; a monolithic SNARK or IVC avoids proof composition but reintroduces expensive non-native signature verification inside the circuit; pure Σ-protocol approaches are efficient for algebraic statements

but do not support arbitrary predicate circuits; and stateful reuse without function hiding leaks the recursive update trace. SHIFT combines proof composition with function-hiding IVC to satisfy these requirements together.

Definitional gap. Stateful reuse introduces a linkage channel that does not arise in stateless presentations. Even if each token hides the holder's credentials and attributes, two tokens derived from the same evolving proof state may reveal that they share a reuse history. For example, a verifier observing two presentation tokens may not learn the credential identity, but may still infer that the tokens were obtained by applying the same sequence of update and rerandomization operations to a common prior state. Existing unlinkability definitions compare independent presentations and therefore do not capture this flow-mode leakage. SHIFT formalizes this setting by distinguishing fresh-mode unlinkability, which compares independent reuse flows, from flow-mode unlinkability, which compares tokens from the same reuse chain against independently generated tokens under the same public schedule.

Our approach. SHIFT addresses both the efficiency and the definitional gaps. Our security model introduces a two-mode unlinkability game: in *fresh mode*, the adversary tries to link two tokens from independent reuse flows; in *flow mode*, the adversary holds two tokens from the *same* reuse flow and tries to determine whether they originate from the same credential or from two distinct credentials. Flow-mode unlinkability is specific to the stateful reuse setting; it captures a linkage channel not addressed by standard stateless presentation definitions and models colluding verifiers who pool session transcripts.

Achieving flow-mode unlinkability requires hiding not just credential identity but also the sequence of update operations performed at each step. We address this with SUPERHALO, a non-uniform incrementally verifiable computation (IVC) scheme with a *function-hiding* property: an adversary holding consecutive proof states cannot determine which predicate function was applied at which step. This property is orthogonal to zero-knowledge (which hides witness values, not function identities). The standard public-state interfaces of Nova [15], SuperNova [16], and HyperNova [17] do not hide this trace by themselves: their accumulator state can reveal which transition is active unless additional hiding machinery is added.

Our second contribution addresses the per-session prover bottleneck for legacy signatures. SHIFT designs a rerandomizable proof of possession (PoP) for BLS and ECDSA signatures using proof composition [13], [14]: expensive pairing and group-exponentiation operations are proved outside the recursive circuit via an efficient Σ-protocol, while the IVC binds the resulting proof to the session context and enables rerandomization by updating the Fiat–Shamir challenge. After the first full presentation, a holder rerandomizes or updates incrementally without re-proving the full signature-verification relation inside the SNARK. Rerandomization costs roughly 2^9 constraints and completes in ≈ 0.65 s regardless of attribute or predicate size; incremental predicate update achieves up to $3.5\times$ savings over full re-proving at large predicate sizes.

Our main contributions are as follows.

- **Reusable anonymous-credential model.** We formalize reusable anonymous credentials as a stateful presentation primitive with two holder-side operations: rerandomization, which derives a fresh token for the same accumulated predicates, and update, which appends a new predicate proof. The security model includes correctness, unforgeability, anonymity, and unlinkability. The unlinkability game has two modes: fresh mode captures linkage across independent reuse flows, while flow mode captures linkage between tokens derived from the same reuse chain.
- **SUPERHALO.** We construct SUPERHALO, a function-hiding multi-function IVC based on Halo-style accumulation with Kate–Zaverucha–Goldberg (KZG) polynomial commitments [18]. In SuperNova-style multi-function IVC interfaces, each supported function has its own accumulator slot and only the active slot changes at each step, so consecutive public proof states can reveal which function was applied. SUPERHALO instead updates every slot at every step, using dummy transitions for inactive functions. This uniform update pattern hides the active transition and prevents the reusable proof state from leaking the update trace needed for flow-mode linking.
- **Rerandomizable proof of possession.** We construct a rerandomizable proof of possession (PoP) for legacy BLS credentials by composing an external Σ-protocol with the recursive proof. The external proof handles expensive legacy-signature algebra, including BLS pairings, while the recursive circuit checks digest binding, rerandomization consistency, predicates, and session context. This keeps pairing and non-native signature verification out of the recursive circuit and lets later presentations rerandomize or update the state without re-proving the full credential-signature relation.
- **Implementation and evaluation.** We implement SHIFT for BLS credentials and an ECDSA variant without the rerandomizable PoP optimization. Rerandomization takes about 0.65s, independent of attribute and predicate size; proof size is below 2KB; and verification is below 50ms. Incremental predicate update reduces prover time by up to $3.5\times$ against the Woo *et al.* baseline in the evaluated setting.

The rest of the paper is organized as follows. Section 2 provides an overview of SHIFT. Section 3 formally defines reusable anonymous credentials. Section 4 introduces the building blocks, and Section 5 presents our construction. Section 6 summarizes the evaluation. We review related work in Section 7.

2. Overview

2.1. System Workflow

In SHIFT, an issuer $\mathcal{I}$ attests to the (identity) attributes of a holder $\mathcal{H}$ by issuing a *verifiable credential* $\mathsf{cred} := (\vec{a}, \sigma)$,

consisting of an attribute vector $\vec{a}$ and a digital signature σ under the issuer's signing key sk. After issuance, $\mathcal{H}$ interacts with a verifier $\mathcal{V}$ to access services by producing a *presentation token* pt. A token proves (i) credential validity under verification key vk of $\mathcal{I}$ and (ii) access-control predicate ϕ of $\mathcal{V}$ holds over $\vec{a}$, without revealing $\vec{a}$ or linking sessions.

SHIFT additionally supports two stateful operations on an existing presentation state S:

- *Rerandomization* (Rerand): derives a fresh, unlinkable token for the same predicate, amortizing the dominant proving cost.
- *Incremental update* (Update): appends a new predicate proof to an existing state without re-proving the full signature-verification relation.

Both operations produce tokens that are unlinkable to all prior sessions, including tokens from the same reuse flow.

2.2. Security Model

We assume an honest issuer $\mathcal{I}$ who correctly attests holder attributes and does not collude with other parties. We consider a PPT adversary $\mathcal{A}$ that may corrupt holders and attempt to produce an accepting token without possessing a valid credential satisfying the required predicates. We further consider semi-honest verifiers that follow the verification protocol faithfully but attempt to link multiple sessions to the same holder or credential. Specifically, we model collusion among verifiers: an adversary controlling any subset of verifiers may pool all observed tokens and attempt to break unlinkability.

We state the four security goals informally here; formal game-based definitions appear in Section 3.

Correctness. An honest holder with a valid credential satisfying the verifier's predicates always produces a token accepted by an honest verifier. Correctness must hold for any reuse schedule: after any sequence of Present, Update, and Rerand calls, each resulting token verifies for the accumulated predicate list and the current session context.

Unforgeability. No PPT adversary can produce an accepting token $(\mathsf{pt}^*, \phi_1^*, \ldots, \phi_n^*, \mathsf{ctx}^*)$ without holding a credential whose attributes satisfy all ϕ_j^*, even after adaptively obtaining honestly generated tokens via the reuse oracles.

Anonymity. A presentation token reveals no information about the holder's identity or attribute values beyond what is logically implied by the proved predicates. In particular, given two candidate credentials whose attribute vectors both satisfy the public predicate, no PPT distinguisher can determine which credential produced the token.

Unlinkability. No PPT adversary can determine whether two tokens originate from the same underlying credential. Unlinkability captures two distinct leakage scenarios:

- *Fresh mode*: the two tokens are derived from independent reuse flows, possibly for different credentials. Unlinkability requires that cross-flow tokens are indistinguishable from same-credential tokens.
- *Flow mode*: the two tokens are derived from the *same* reuse flow of one credential. Unlinkability requires that intra-flow tokens are indistinguishable from tokens generated from two independent credentials under the same public schedule.

Flow-mode unlinkability is specific to reusable AC systems and is not captured by any prior unlinkability definition; it ensures that a verifier colluding across sessions cannot detect that two tokens belong to the same reuse chain.

2.3. Technical Overview

A presentation token pt is bound to a session-specific context (*e.g.*, a nonce) to prevent replay and to eliminate trivial linkability. Generating pt from scratch is prover-intensive: verifying a legacy signature (ECDSA, BLS) inside a zk-SNARK circuit costs 2^{18}–2^{22} constraints, yielding proving times of several seconds per session.

SHIFT models presentation reuse as a *non-uniform incrementally verifiable computation* (NIVC) problem: the holder maintains a compact running proof π_i and a state S, and derives a fresh token by applying a lightweight rerandomize or update step, appending only the cost of the new predicate. The proof size stays $O(1)$, and the update cost depends only on the newly appended function.

Directly instantiating a standard public-state NIVC scheme (*e.g.*, SuperNova [16]) is insufficient for two reasons:

(1) Trace leakage breaks unlinkability. Standard SuperNova-style NIVC constructions update only the idx-th instance-witness slot when evaluating a function F_{idx}. By component-wise comparison of accumulator vectors across consecutive proofs, an adversary can deduce idx without opening any commitments. This structural leakage exposes the internal transition trace (*e.g.*, shared reuse prefixes), immediately breaking flow-mode unlinkability.

To eliminate this structural leakage, SHIFT introduces SUPERHALO (Section 4.1), which enforces *uniform dummy updates*. At each step, alongside the evaluation of F_{idx}, the prover executes a trivial dummy aggregation on the remaining $\ell - 1$ slots. Consequently, all ℓ entries of $\vec{\mathsf{x}}$ receive fresh public updates at every step, reducing the structural part of function hiding to the hiding property of the underlying KZG commitment scheme. The per-step overhead is strictly ℓ PCS aggregations; for our access control application, where $\ell = 2 + |L_\phi|$ (one slot for F_{PoP}, one for F_{re}, one per predicate type), this cost remains small and tightly bounded.

A remark on scope: Function hiding (concealing *which* function from $\{F_1, \ldots, F_\ell\}$ was applied) is orthogonal to *iteration hiding* (concealing the total *number* of IVC steps), as explored in ALPACA [19]. SHIFT targets both properties: the former via uniform dummy updates, and the latter via the zero-knowledge and hiding properties of the underlying KZG accumulation. Function hiding explicitly conceals the internal transition trace of the reusable proof state (*i.e.*, the invocation sequence of F_{pop}, F_{re}, and $F_\phi : \phi \in L_\phi$) to prevent flow-mode linking. However, it does not obscure the fi-

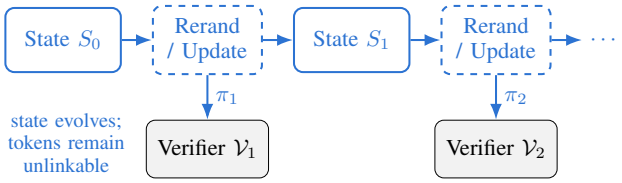


Figure 1: Sketch of SHIFT: A holder maintains a compact state S_t and derives unlinkable presentations by F_{re} or F_{ϕ_i}.

nal attested policy, as the predicate labels $\Phi = (\phi_1, \ldots, \phi_n)$ supplied to Verify remain public.

(2) Legacy signature verification inside circuits is expensive. Proving a pairing-based signature (BLS) inside a SNARK circuit requires emulating non-native group operations, costing roughly $2 \cdot 2^{22}$ constraints for a single pairing. Proving an ECDSA signature over P-256 requires 2^{18}–2^{20} constraints per exponentiation.

SHIFT addresses this with a *rerandomizable proof of possession* (PoP) for BLS signatures, constructed via *proof composition* [13], [14] (Section 4.2). The expensive pairing checks are moved outside the recursive circuit and proved via an efficient Σ-protocol over ElGamal commitments, while the IVC binds the resulting proof to the presentation context and enables rerandomization.

Finally, to align with the reusable AC syntax, we define in Section 5.1 the function family handled by SUPERHALO:

$$\mathcal{F} = \{F_{\mathsf{pop}}, F_{\mathsf{re}}\} \cup \{F_\phi : \phi \in L_\phi\}.$$

Here F_{pop} initializes the credential proof-of-possession state, F_{re} rerandomizes the presentation state for a fresh session context, and F_ϕ checks the predicate $\phi(\vec{a}) = 1$.

In the full construction (Section 5.2), a holder computes F_{PoP} once at initial presentation, selectively appends F_{ϕ_i} steps for each required predicate, and finalizes with F_{re} to rerandomize the running proof before producing the presentation token. Subsequent rerandomizations cost only one F_{re} step regardless of attribute or predicate size (see Figure 1).

3. Reusable Anonymous Credential Scheme

In this section, we provide the formal definition of the *reusable anonymous credential* scheme, including its syntax and security properties.

3.1. Syntax

We define the syntax of a reusable anonymous credential scheme. In contrast to prior formulations [20], [12], our definition additionally supports rerandomization and predicate updates.

Definition 1 (Reusable Anonymous Credential Scheme)**.** *A reusable anonymous credential scheme is a tuple of PPT algorithms:*

$$\Omega = \left(\begin{array}{c} \mathsf{Setup},\ \mathsf{IKeyGen},\ \mathsf{HKeyGen},\ \mathsf{Issue}, \\ \mathsf{Present},\ \mathsf{Update},\ \mathsf{Rerand},\ \mathsf{Verify} \end{array} \right),$$

with the following syntax.

- $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*: outputs public parameters* pp.
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{IKeyGen}(\mathsf{pp})$*: outputs the issuer key pair* $(\mathsf{sk}, \mathsf{vk})$.
- $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow\!\!\$\ \mathsf{HKeyGen}(\mathsf{pp}, L_\phi)$*: outputs holder proof keys for the supported predicate list* L_ϕ*. The verifying key* vk_Π *encodes* L_ϕ *so that* Verify *can check which predicates a token covers.*
- $\mathsf{cred} \leftarrow \mathsf{Issue}(\mathsf{pp}, \vec{a}, \mathsf{sk})$*: issues a credential* cred *for attribute vector* $\vec{a}$ *under* sk.
- $(\mathsf{pt}, S) \leftarrow \mathsf{Present}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}, \mathsf{cred}, \phi, \mathsf{ctx})$*: for an initial predicate* $\phi \in L_\phi$ *and context* ctx*, outputs a token* pt *and state* S.
- $(\mathsf{pt}, S') \leftarrow \mathsf{Update}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \phi, \mathsf{ctx})$*: appends predicate* $\phi \in L_\phi$ *to state* S *and outputs an updated token* pt *and state* S'.
- $(\mathsf{pt}, S') \leftarrow \mathsf{Rerand}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \mathsf{ctx})$*: rerandomizes state* S *for context* ctx *and outputs a fresh token* pt *and state* S'.
- $b \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}, \phi_1, \ldots, \phi_n, \mathsf{ctx})$*: checks token* pt *against predicates* $\phi_1, \ldots, \phi_n \in L_\phi$ *and context* ctx*, and outputs an acceptance bit* b.

Remark 1 (State and presentable predicates)**.** *The state* S *is a holder-side secret: It is never sent to the verifier and never exposed by* Verify*. The predicates* $\phi_1, \ldots, \phi_n$ *supplied to* Verify *are* public labels *known to the verifier; They describe the set of predicates whose proofs are accumulated in* pt *over the reuse flow.*

3.2. Security Definitions

We formalize the security of a reusable anonymous credential scheme via standard game-based experiments. All experiments use auxiliary procedures Good and Run from Fig. 2. The procedure Good checks that a public reuse schedule τ is well-formed (*i.e.*, the first command is Present, each subsequent command is Update or Rerand, and every predicate in the schedule is in L_ϕ and satisfied by the candidate attribute vector). The procedure Run executes the schedule faithfully and returns the tokens at the challenged positions.

Definition 2 (Correctness)**.** *Let* Ω *be a reusable anonymous credential scheme. Consider the experiment* $\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda)$ *defined in Fig. 3. We say that* Ω *satisfies* correctness *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\Pr\left[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.

Definition 3 (Unforgeability)**.** *Let* Ω *be a reusable anonymous credential scheme. Consider the experiment* $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{A}}(\lambda)$ *defined in Fig. 5, where* $\mathcal{A}$ *may adaptively query the oracles in Fig. 4. We define the unforgeability advantage of* $\mathcal{A}$ *against* Ω *as* $\mathsf{Adv}^{\mathsf{uf}}_{\Omega}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{A}}(\lambda) = 1\right]$*. We say that* Ω *satisfies* unforgeability *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{uf}}_{\Omega}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Definition 4 (Anonymity)**.** *Let* Ω *be a reusable anonymous credential scheme. Consider the experiment* $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda)$ *defined in Fig. 6. We define the anonymity advantage of* $\mathcal{A}$

$\mathsf{Good}(L_\phi, \vec{a}, \tau, I)$:

Parse $\tau = (T, (\mathsf{cmd}_t)_{t\in[T]})$;
If $T < 1 \vee I \not\subseteq [T]$ then return 0;
For $t \in [T]$:
  If $\mathsf{cmd}_t = \langle \mathsf{Present}, \phi, \mathsf{ctx}\rangle$ or $\langle \mathsf{Update}, \phi, \mathsf{ctx}\rangle$:
    If $\phi \notin L_\phi \vee \phi(\vec{a}) \neq 1$ then return 0;
Return 1.

$\mathsf{Run}(\mathsf{cred}, \tau, I)$:

Parse $\tau = (T, (\mathsf{cmd}_t)_{t\in[T]})$;
$S \leftarrow \bot$;
For $t \in [T]$:
  Parse cmd_t as $\langle \mathsf{Present}, \phi, \mathsf{ctx}\rangle$, $\langle \mathsf{Update}, \phi, \mathsf{ctx}\rangle$,
  or $\langle \mathsf{Rerand}, \mathsf{ctx}\rangle$;
  If $\mathsf{cmd}_t = \langle \mathsf{Present}, \phi, \mathsf{ctx}\rangle$:
    If $S \neq \bot$ then return $\bot$;
    $(\mathsf{pt}, S) \leftarrow \Omega.\mathsf{Present}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}, \mathsf{cred}, \phi, \mathsf{ctx})$;
  If $\mathsf{cmd}_t = \langle \mathsf{Update}, \phi, \mathsf{ctx}\rangle$:
    If $S = \bot$ then return $\bot$;
    $(\mathsf{pt}, S) \leftarrow \Omega.\mathsf{Update}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \phi, \mathsf{ctx})$;
  If $\mathsf{cmd}_t = \langle \mathsf{Rerand}, \mathsf{ctx}\rangle$:
    If $S = \bot$ then return $\bot$;
    $(\mathsf{pt}, S) \leftarrow \Omega.\mathsf{Rerand}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \mathsf{ctx})$;
  If $t \in I$ then $\mathsf{pt}^*_t \leftarrow \mathsf{pt}$;
Return $(\mathsf{pt}^*_t)_{t\in I}$.

Figure 2: Auxiliary procedures.

$\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda)$:

$\mathsf{pp} \leftarrow\!\$\ \Omega.\mathsf{Setup}(1^\lambda)$;
$(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\$\ \Omega.\mathsf{IKeyGen}(\mathsf{pp})$;
$(\vec{a}, L_\phi, \tau) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk})$;
$(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow \Omega.\mathsf{HKeyGen}(\mathsf{pp}, L_\phi)$;
$\mathsf{cred} \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}, \mathsf{sk})$;
Parse $\tau = (T, (\mathsf{cmd}_t)_{t\in[T]})$;
If $\neg\mathsf{Good}(L_\phi, \vec{a}, \tau, [T])$ then return 0;
$(\mathsf{pt}_t)_{t\in[T]} \leftarrow \mathsf{Run}(\mathsf{cred}, \tau, [T])$;
If $\exists\, t \in [T]$ such that $\mathsf{pt}_t = \bot$ then return 0;
$\Phi \leftarrow \emptyset$;
For $t \in [T]$:
  Parse cmd_t as $\langle \mathsf{Present}, \phi, \mathsf{ctx}\rangle$, $\langle \mathsf{Update}, \phi, \mathsf{ctx}\rangle$,
  or $\langle \mathsf{Rerand}, \mathsf{ctx}\rangle$;
  If $\mathsf{cmd}_t = \langle \mathsf{Present}, \phi, \mathsf{ctx}\rangle$ or $\langle \mathsf{Update}, \phi, \mathsf{ctx}\rangle$:
    Then $\Phi \leftarrow (\Phi, \phi)$;
  If $\Omega.\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}_t, \Phi, \mathsf{ctx}) \neq 1$ then return 1;
Return 0.

Figure 3: The correctness experiment.

against Ω *as* $\mathsf{Adv}^{\mathsf{anon}}_\Omega(\mathcal{A}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$. *We say that* Ω *satisfies* anonymity *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{anon}}_\Omega(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Definition 5 (Unlinkability)**.** *Let* Ω *be a reusable anonymous credential scheme. Consider the experiment* $\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda)$ *defined in Fig. 7. We define the unlinkability advantage of* $\mathcal{A}$ *against* Ω *as* $\mathsf{Adv}^{\mathsf{unlink}}_\Omega(\mathcal{A}) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$. *We say that* Ω *satisfies* unlinkability *if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{unlink}}_\Omega(\mathcal{A}) \leq \mathsf{negl}(\lambda)$.

Remark 2 (Anonymity versus unlinkability)**.** *The two games measure distinct notions of leakage, and neither implies the other. Anonymity captures* credential identity hiding*: Given a single token, the adversary cannot determine which of two candidate credentials produced it. Unlinkability captures* session linkage hiding*: Given two tokens, the adversary cannot determine whether they originate from the same or from different credentials. Flow-mode unlinkability goes further: It captures whether two tokens from the* same *reuse flow (sharing a common state prefix) leak that they are related. Neither anonymity nor fresh-mode unlinkability implies flow-mode unlinkability.*

4. Building Blocks

This section introduces the building blocks for SHIFT. We construct a function-hiding Non-uniform Incrementally Verifiable Computation (NIVC) scheme in Section 4.1, and a rerandomizable proof of possession (PoP) for BLS signatures in Section 4.2.

4.1. Function-Hiding Incrementally Verifiable Computation

Recall the presentation-reuse workflow in Section 2.3. Intuitively, this process is modeled as an NIVC problem [21], [16].

We use "non-uniform" in the SuperNova sense: the computation is a sequence of steps, where each step applies one function from a fixed family $\mathcal{F} = \{F_1, \ldots, F_\ell\}$ rather than a single universal transition function. The additional requirement in our setting is that the active function index must remain hidden, *i.e.*, *function-hiding* property.

Definition 6 (Function-Hiding NIVC)**.** *A function-hiding NIVC scheme for a function set* $\{F_1, \ldots, F_\ell\}$ *is defined as a tuple* $\mathsf{NIVC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$*: on input security parameter* 1^λ*, outputs public parameters* pp*, including the maximum iteration step* N*.*
- $(\mathsf{pk}, \mathsf{vk}) \leftarrow\!\$\ \mathsf{KeyGen}(\mathsf{pp}, (F_1, \ldots, F_\ell))$*: on input* pp *and function set* $(F_1, \ldots, F_\ell)$*, outputs a key pair* $(\mathsf{pk}, \mathsf{vk})$*.*
- $\pi_{i+1} \leftarrow \mathsf{Prove}(\mathsf{pk}, (z_0, z_i), (i, \pi_i, \mathsf{w}_i, \mathsf{idx}))$*: on input a prover key* pk*, initial input* z_0*, claimed output* z_i*, an NIVC proof* π_i*, a non-deterministic witness* w_i*, and a function index* idx*, produces a new proof* π_{i+1} *attesting to* $z_{i+1} = F_{\mathsf{idx}}(z_i, \mathsf{w}_i)$*.*
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, (z_0, z_i), \pi_i)$*: on input a verifier key* vk*, initial input* z_0*, claimed output* z_i*, and an NIVC proof* π_i*, outputs* $b = 1$ *if* π_i *is accepted and* $b = 0$ *otherwise.*

Consider the games in Fig. 8. We say that NIVC *is* function-hiding *if for every PPT adversary* $\mathcal{A}$

$\mathcal{O}_{\mathsf{Issue}}(\vec{a})$:

1 : $\mathsf{cred} \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}, \mathsf{sk})$;
2 : Pick fresh handle i;
3 : Set $\mathcal{Q}_{\mathsf{Issue}}[i] \leftarrow (\mathsf{cred}, \vec{a})$;
4 : Return i.

$\mathcal{O}_{\mathsf{Reveal}}(i)$:

1 : Parse $\mathcal{Q}_{\mathsf{Issue}}[i] = (\mathsf{cred}, \vec{a})$;
2 : Add $(\mathsf{cred}, \vec{a})$ to $\mathcal{Q}_{\mathsf{Rev}}$;
3 : Return cred.

$\mathcal{O}_{\mathsf{Present}}(i, \phi, \mathsf{ctx})$:

1 : Parse $\mathcal{Q}_{\mathsf{Issue}}[i] = (\mathsf{cred}, \cdot)$;
2 : $(\mathsf{pt}, S) \leftarrow \Omega.\mathsf{Present}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}, \mathsf{cred}, \phi, \mathsf{ctx})$;
3 : Pick fresh h and set $\mathcal{T}[h] \leftarrow (i, S)$;
4 : Add pt to $\mathcal{Q}_{\mathsf{Tok}}$ and return (pt, h).

$\mathcal{O}_{\mathsf{Update}}(h, \phi, \mathsf{ctx})$:

1 : Parse $\mathcal{T}[h] = (i, S)$;
2 : $(\mathsf{pt}, S') \leftarrow \Omega.\mathsf{Update}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \phi, \mathsf{ctx})$;
3 : Set $\mathcal{T}[h] \leftarrow (i, S')$ and add pt to $\mathcal{Q}_{\mathsf{Tok}}$;
4 : Return pt.

$\mathcal{O}_{\mathsf{Rerand}}(h, \mathsf{ctx})$:

1 : Parse $\mathcal{T}[h] = (i, S)$;
2 : $(\mathsf{pt}, S') \leftarrow \Omega.\mathsf{Rerand}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \mathsf{ctx})$;
3 : Set $\mathcal{T}[h] \leftarrow (i, S')$ and add pt to $\mathcal{Q}_{\mathsf{Tok}}$;
4 : Return pt.

Figure 4: The oracles that the adversary can access. $\mathcal{Q}_{\mathsf{Issue}}$ is a challenger table with $\mathcal{Q}_{\mathsf{Issue}}[i] = (\mathsf{cred}, \vec{a})$ for issuance handle i; $\mathcal{Q}_{\mathsf{Rev}}$ collects revealed pairs $(\mathsf{cred}, \vec{a})$; $\mathcal{T}$ maps a state handle h to (i, S) where S is the hidden reuse state; and $\mathcal{Q}_{\mathsf{Tok}}$ records all tokens pt output by the holder oracles.

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{A}}(\lambda)$:

1 : $\mathsf{pp} \leftarrow\!\!\$\ \Omega.\mathsf{Setup}(1^\lambda)$;
2 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \Omega.\mathsf{IKeyGen}(\mathsf{pp})$;
3 : $L_\phi \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk})$;
4 : $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow \Omega.\mathsf{HKeyGen}(\mathsf{pp}, L_\phi)$;
5 : $\mathcal{Q}_{\mathsf{Issue}} \leftarrow \emptyset,\ \mathcal{Q}_{\mathsf{Rev}} \leftarrow \emptyset,\ \mathcal{Q}_{\mathsf{Tok}} \leftarrow \emptyset$;
6 : Maintain a map $\mathcal{T}$ from handles h to states S;
7 : $\mathcal{O} = \{\mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Reveal}}, \mathcal{O}_{\mathsf{Present}}, \mathcal{O}_{\mathsf{Update}}, \mathcal{O}_{\mathsf{Rerand}}\}$;
8 : $r^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}_\Pi, \mathsf{vk}, L_\phi)$;
9 : Parse r^* as $(\mathsf{pt}^*, (\phi^*_1, \ldots, \phi^*_n), \mathsf{ctx}^*)$;
10 : If $n < 1$ then return 0;
11 : If $\exists\, j \in \{1, \ldots, n\}$ s.t. $\phi^*_j \notin L_\phi$ then return 0;
12 : Return 1 if :
13 : $\quad \Omega.\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}^*, \phi^*_1, \ldots, \phi^*_n, \mathsf{ctx}^*) = 1$;
14 : $\quad \mathsf{pt}^* \notin \mathcal{Q}_{\mathsf{Tok}}$;
15 : $\quad$ For each $(\mathsf{cred}_i, \vec{a}_i) \in \mathcal{Q}_{\mathsf{Rev}}$,
16 : $\quad\quad \exists\, j \in \{1, \ldots, n\}$ such that $\phi^*_j(\vec{a}_i) \neq 1$;
17 : Return 0.

Figure 5: The unforgeability experiment.

$\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda)$:

1 : $\mathsf{pp} \leftarrow\!\!\$\ \Omega.\mathsf{Setup}(1^\lambda)$;
2 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \Omega.\mathsf{IKeyGen}(\mathsf{pp})$;
3 : $L_\phi \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk})$;
4 : $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow \Omega.\mathsf{HKeyGen}(\mathsf{pp}, L_\phi)$;
5 : $\mathcal{Q}_{\mathsf{Issue}}, \mathcal{Q}_{\mathsf{Tok}}, \mathcal{T} \leftarrow \emptyset$;
6 : $\mathcal{O} = \{\mathcal{O}_{\mathsf{Present}}, \mathcal{O}_{\mathsf{Update}}, \mathcal{O}_{\mathsf{Rerand}}\}$;
7 : $(\vec{a}_0, \vec{a}_1, \tau, s) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}_\Pi, \mathsf{vk}, L_\phi)$;
8 : $\mathsf{cred}_0 \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}_0, \mathsf{sk})$;
9 : $\mathsf{cred}_1 \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}_1, \mathsf{sk})$;
10 : Set $\mathcal{Q}_{\mathsf{Issue}}[0] \leftarrow (\mathsf{cred}_0, \vec{a}_0)$ and $\mathcal{Q}_{\mathsf{Issue}}[1] \leftarrow (\mathsf{cred}_1, \vec{a}_1)$;
11 : If $\neg\mathsf{Good}(L_\phi, \vec{a}_0, \tau, \{s\}) \vee \neg\mathsf{Good}(L_\phi, \vec{a}_1, \tau, \{s\})$
12 : $\quad$ Then return $b' \leftarrow\!\!\$\ \{0, 1\}$;
13 : $b \leftarrow\!\!\$\ \{0, 1\}$;
14 : $\mathsf{pt} \leftarrow \mathsf{Run}(\mathsf{cred}_b, \tau, \{s\})$;
15 : If $\mathsf{pt} = \perp$ then return $b' \leftarrow\!\!\$\ \{0, 1\}$;
16 : $b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pt})$;
17 : Return $[b^* = b]$.

Figure 6: The anonymity experiment.

there exists a PPT (stateful) simulator $\mathcal{S}$ such that $\left|\Pr\left[\mathsf{NIVC\text{-}PH}^{\mathcal{A}}_{0,\mathsf{NIVC}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{NIVC\text{-}PH}^{\mathcal{A},\mathcal{S}}_{1,\mathsf{NIVC}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$.

We use iteration hiding and function hiding as two separate requirements. Iteration hiding, as used in ALPACA [19], hides how many times the updatable proof relation has been applied. Function hiding is orthogonal: for a fixed number of steps, it hides which function indices from $\{F_1, \ldots, F_\ell\}$ were applied at those steps. Neither property implies the other in general.

Despite this function-hiding property, constructing a practical function-hiding NIVC scheme is non-trivial, as existing implementations fail to satisfy the following requirements.

- *Constant proof size*: for succinct verification, SHIFT uses a constant-sized polynomial commitment scheme, specifically KZG [18].
- *Fast proof generation*: unlike IVC applications that prove long computation sequences offline, an AC system requires fast per-session proof generation.
- *Zero knowledge*: presentation proofs must be zero-knowledge, and a credential verifier must not be able to distinguish a freshly generated presentation from a rerandomized one. The latter is ensured by the *iteration-hiding* property of the IVC scheme [19].

Guided by these criteria, we build our function-hiding NIVC on a Halo-style recursive construction [22], [23], which offers faster proof generation than Nova-style approaches [15], [16], [24] that require a separate outer SNARK to compress the accumulator. For polynomial commitments, we instantiate the scheme with the hiding variant of KZG [18], yielding constant-size proofs and efficient verification. We denote the resulting construction by SUPERHALO (Construction 1). Appendix B gives the security analysis and additional details.

$\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda)$:

1 : $\mathsf{pp} \leftarrow\!\$\ \Omega.\mathsf{Setup}(1^\lambda)$;
2 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\$\ \Omega.\mathsf{IKeyGen}(\mathsf{pp})$;
3 : $(L_\phi, \vec{a}_0, \vec{a}_1, \mathsf{mode}, \mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk})$;
4 : $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow \Omega.\mathsf{HKeyGen}(\mathsf{pp}, L_\phi)$;
5 : $\mathsf{cred}_0 \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}_0, \mathsf{sk})$;
6 : $\mathsf{cred}_1 \leftarrow \Omega.\mathsf{Issue}(\mathsf{pp}, \vec{a}_1, \mathsf{sk})$;
7 : $b \leftarrow\!\$\ \{0,1\}$;
8 : If mode $=$ fresh then:
9 : Parse $\mathsf{aux} = (\tau_1, s_1, \tau_2, s_2)$;
10 : For $i \in \{0,1\}$ and $j \in \{1,2\}$:
11 : If $\neg\mathsf{Good}(L_\phi, \vec{a}_i, \tau_j, \{s_j\})$ Then return $b' \leftarrow\!\$\ \{0,1\}$;
12 : $\mathsf{pt}_1 \leftarrow \mathsf{Run}(\mathsf{cred}_0, \tau_1, \{s_1\})$;
13 : $\mathsf{pt}_2 \leftarrow \mathsf{Run}(\mathsf{cred}_b, \tau_2, \{s_2\})$;
14 : else if mode $=$ flow then:
15 : Parse $\mathsf{aux} = (\tau, s_1, s_2)$;
16 : If $s_1 = s_2$ then return $b' \leftarrow\!\$\ \{0,1\}$;
17 : For $i \in \{0,1\}$:
18 : If $\neg\mathsf{Good}(L_\phi, \vec{a}_i, \tau, \{s_1, s_2\})$ then return $b' \leftarrow\!\$\ \{0,1\}$;
19 : If $b = 0$ then:
20 : $(\mathsf{pt}_1, \mathsf{pt}_2) \leftarrow \mathsf{Run}(\mathsf{cred}_0, \tau, \{s_1, s_2\})$;
21 : else:
22 : $\mathsf{pt}_1 \leftarrow \mathsf{Run}(\mathsf{cred}_0, \tau, \{s_1\})$;
23 : $\mathsf{pt}_2 \leftarrow \mathsf{Run}(\mathsf{cred}_1, \tau, \{s_2\})$;
24 : Else return $b' \leftarrow\!\$\ \{0,1\}$;
25 : If $\mathsf{pt}_1 = \bot \vee \mathsf{pt}_2 = \bot$ then return $b' \leftarrow\!\$\ \{0,1\}$;
26 : $b^* \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}_1, \mathsf{pt}_2)$;
27 : Return $[b^* = b]$.

Figure 7: The unlinkability experiment. The adversary receives the two challenge tokens $(\mathsf{pt}_1, \mathsf{pt}_2)$ simultaneously and outputs a guess without further oracle access. This is a deliberate simplification: the two tokens are issued jointly, so adaptive oracle queries between the two challenges do not arise naturally. The security proof in Appendix D goes through with this definition.

Game $\mathsf{NIVC\text{-}PH}^{\mathcal{A}}_{0,\mathsf{NIVC}}(1^\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, (F_1, \ldots, F_\ell))$
3 : $b \leftarrow \mathcal{A}^{\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{vk}, \cdot)}(\mathsf{pk}, \mathsf{vk})$
4 : **return** b

Game $\mathsf{NIVC\text{-}PH}^{\mathcal{A},\mathcal{S}}_{1,\mathsf{NIVC}}(1^\lambda)$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, (F_1, \ldots, F_\ell))$
3 : $b \leftarrow \mathcal{A}^{\tilde{\mathcal{S}}(\mathsf{vk}, \cdot)}(\mathsf{pk}, \mathsf{vk})$
4 : **return** b

Oracle $\tilde{\mathcal{P}}(\mathsf{pk}, \mathsf{vk}, z_0, z_i, i, \pi_i, \mathsf{w}_i, \mathsf{idx})$

1 : **if** $\mathsf{idx} \notin [\ell]$
2 : **return** $\bot$
3 : **if** $1 \neq \mathsf{Verify}(\mathsf{vk}, (z_0, z_i), \pi_i)$
4 : **return** $\bot$
5 : $\pi_{i+1} \leftarrow \mathsf{Prove}(\mathsf{pk}, (z_0, z_i), (i, \pi_i, \mathsf{w}_i, \mathsf{idx}))$
6 : **return** π_{i+1}

Oracle $\tilde{\mathcal{S}}(\mathsf{vk}, z_0, z_i, i, \pi_i, \mathsf{w}_i, \mathsf{idx})$

1 : **if** $\mathsf{idx} \notin [\ell]$
2 : **return** $\bot$
3 : **if** $1 \neq \mathsf{Verify}(\mathsf{vk}, (z_0, z_i), \pi_i)$
4 : **return** $\bot$
5 : $z_{i+1} \leftarrow F_{\mathsf{idx}}(z_i, \mathsf{w}_i)$
6 : $\pi_{i+1} \leftarrow \mathcal{S}(\mathsf{vk}, (F_1, \ldots, F_\ell), z_0, z_i, z_{i+1}, i)$
7 : **return** π_{i+1}

Figure 8: Security game for function hiding of NIVC schemes.

Construction 1 (SUPERHALO: Function-Hiding NIVC)**.** *Let* Commit *be a commit algorithm for the polynomial commitment scheme KZG [18] with hiding property and an efficient non-interactive aggregate scheme* Aggregate *involving a prover and a verifier. Each function F from the function set $\{F_1, \ldots, F_\ell\}$ is represented in the Plonkish constraint system [25]. Let $(\mathsf{x}_\bot, \mathsf{w}_\bot)$ denote a trivial committed Plonkish instance-witness pair valid for any represented function. We define* $\mathsf{NIVC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *as follows.*

- $\Pi.\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: Takes the security parameters as input and outputs public parameters for KZG, the aggregate protocol, and* $\mathsf{NIZK}_{\mathsf{plonk}}$.
- $\Pi.\mathsf{KeyGen}(\mathsf{pp}, (F_1, \ldots, F_\ell)) \to (\mathsf{pk}, \mathsf{vk})$*: Takes the public parameters and the function set as input, runs KZG's key generation to produce* $(\mathsf{pk}_{\mathsf{KZG}}, \mathsf{vk}_{\mathsf{KZG}})$*, and for each F_j, a Plonkish key pair* $(\mathsf{pk}_j, \mathsf{vk}_j)$*. Let* $\mathsf{pk} := ((F_j, \mathsf{pk}_j)_{j\in[\ell]}, \mathsf{pk}_{\mathsf{KZG}})$ *and* $\mathsf{vk} := ((F_j, \mathsf{vk}_j)_{j\in[\ell]}, \mathsf{vk}_{\mathsf{KZG}})$.
- $\Pi.\mathsf{Prove}(\mathsf{pk}, (z_0, z_i), (i, \pi_i, \mathsf{w}_i, \mathsf{idx})) \to \pi_{i+1}$*: Generates a proof π_{i+1} that $z_{i+1} = F_{\mathsf{idx}}(z_i, \mathsf{w}_i)$. It first parses π_i as $((\vec{\mathsf{X}}_i, \vec{\mathsf{W}}_i), (\mathsf{x}_i, \mathsf{w}_i))$. If $i = 0$, it initializes $(\mathsf{x}_1, \mathsf{w}_1) := (\mathsf{x}_\bot, \mathsf{w}_\bot)$ and $(\vec{\mathsf{X}}_1, \vec{\mathsf{W}}_1)$ with $\vec{\mathsf{X}}[j] = \mathsf{x}_\bot$ and $\vec{\mathsf{W}}[j] = \mathsf{w}_\bot$ for all $j \in [\ell]$. Otherwise, it updates* all *ℓ slots of $(\vec{\mathsf{X}}_i, \vec{\mathsf{W}}_i)$ as follows:*
  - Active slot *($j = \mathsf{idx}$): run* $(\mathsf{X}_{i+1}, \mathsf{W}_{i+1}, \pi_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}.\mathcal{P}(\mathsf{pk}_{\mathsf{KZG}}, (\vec{\mathsf{X}}_i[\mathsf{idx}], \vec{\mathsf{W}}_i[\mathsf{idx}]), (\mathsf{x}_i, \mathsf{w}_i))$ *and set* $\vec{\mathsf{X}}_{i+1}[\mathsf{idx}] := \mathsf{X}_{i+1}$, $\vec{\mathsf{W}}_{i+1}[\mathsf{idx}] := \mathsf{W}_{i+1}$.
  - Passive slots *($j \in [\ell], j \neq \mathsf{idx}$): run* $(\mathsf{X}_{i+1,j}, \mathsf{W}_{i+1,j}, \pi^j_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}.\mathcal{P}(\mathsf{pk}_{\mathsf{KZG}}, (\vec{\mathsf{X}}_i[j], \vec{\mathsf{W}}_i[j]), (\mathsf{x}_\bot, \mathsf{w}_\bot))$ *and set* $\vec{\mathsf{X}}_{i+1}[j] := \mathsf{X}_{i+1,j}$, $\vec{\mathsf{W}}_{i+1}[j] := \mathsf{W}_{i+1,j}$.

  Then it runs $\mathsf{NIZK}_{\mathsf{plonk}}.\mathsf{Prove}$ *for the relation $\mathcal{R} =$*

$$\left\{ \begin{array}{l} (\mathsf{vk}_{\mathsf{KZG}}, \vec{\mathsf{X}}_i, \mathsf{x}_i, \vec{\mathsf{X}}_{i+1}, (i, z_0, z_i)); \\ (\mathsf{idx}, \mathsf{w}_i, \pi_{\mathsf{agg}}, \{\pi^j_{\mathsf{agg}}\}_{j\neq\mathsf{idx}}): \\ \left(\begin{array}{l} z_{i+1} \leftarrow F_{\mathsf{idx}}(z_i, \mathsf{w}_i) \wedge z_0 = z_i \text{ if } i = 0 \\ \mathsf{x}_i.x = \mathsf{Hash}(\mathsf{vk}_{\mathsf{KZG}}, z_0, z_i, \vec{\mathsf{X}}_i) \\ i \leq N, \quad F_{\mathsf{idx}} \in \{F_1, \ldots, F_\ell\}. \\ \mathsf{Aggregate}.\mathcal{V}(\mathsf{vk}_{\mathsf{KZG}}, (\vec{\mathsf{X}}_i[\mathsf{idx}], \mathsf{x}_i), \vec{\mathsf{X}}_{i+1}[\mathsf{idx}], \pi_{\mathsf{agg}}) = 1. \\ \forall\, j \in [\ell],\ j \neq \mathsf{idx}: \\ \quad \mathsf{Aggregate}.\mathcal{V}(\mathsf{vk}_{\mathsf{KZG}}, (\vec{\mathsf{X}}_i[j], \mathsf{x}_\bot), \vec{\mathsf{X}}_{i+1}[j], \pi^j_{\mathsf{agg}}) = 1. \\ \mathsf{x}_{i+1}.x = \mathsf{Hash}(\mathsf{vk}_{\mathsf{KZG}}, z_0, z_{i+1}, \vec{\mathsf{X}}_{i+1}) \end{array} \right) \end{array} \right\},$$

to obtain $(\mathsf{x}_{i+1}, \mathsf{w}_{i+1})$. *The function index* idx *appears as a* witness *in* $\mathcal{R}$*; the zero-knowledge property of* $\mathsf{NIZK}_{\mathsf{plonk}}$ *therefore hides* idx. *Finally, it outputs* $\pi_{i+1} \coloneqq ((\vec{\mathsf{X}}_{i+1}, \vec{\mathsf{W}}_{i+1}), (\mathsf{x}_{i+1}, \mathsf{w}_{i+1}))$.

- $\Pi.\mathsf{Verify}(\mathsf{vk}, (z_0, z_i), \pi_i) \to b$*: Outputs the verification result. If* $i = 0$*, it checks* $z_i = z_0$*. Otherwise, it parses* π_i *as* $((\vec{\mathsf{X}}_i, \vec{\mathsf{W}}_i), (\mathsf{x}_i, \mathsf{w}_i))$*; checks the public hash binding for* x_i*, verifies the active* $\mathsf{NIZK}_{\mathsf{plonk}}$ *instance, and verifies each cached committed Plonkish pair* $(\vec{\mathsf{X}}_i[j], \vec{\mathsf{W}}_i[j])$ *under* vk_j *for all* $j \in [\ell]$*. It outputs the conjunction of these checks.*

Remark 3 (Implementing passive updates)**.** *The implementation may avoid materializing the full passive witness computations, but it must still sample fresh blinding randomness and output a fresh dummy aggregated instance for every passive slot. Equivalently, the public accumulator vector changes in all* ℓ *slots at every step, while each inactive slot is updated using* $(\mathsf{x}_\perp, \mathsf{w}_\perp)$*. Step 12 in Figure 9 is the auxiliary aggregation that publishes these slot updates; it is not a license to leave passive public entries unchanged. Thus the verifier sees the same update pattern for every slot and cannot identify* idx *by comparing consecutive accumulator vectors.*

4.2. Rerandomizable PoP for Signatures

We formalize a rerandomizability-aware Proof of Possession variant with an efficient construction for BLS signatures via proof composition.

Definition 7 (Rerandomizable PoP for Signatures)**.** *Let* $\Theta = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Check})$ *be a signature scheme and* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *be an NIZK proof system. We define a proof of possession for a signature* σ *as an NIZK proof with respect to the following relation:*

$$\mathcal{R}_{\mathsf{PoPS}} \coloneqq \left\{ \ (vk, \sigma; m) \colon \Theta.\mathsf{Check}(vk, \sigma, m) = 1 \ \right\},$$

where vk *denotes the verification key,* m *denotes the message, and* σ *denotes the signature. Furthermore, if there exists an efficient algorithm that transforms a proof* π *for* $(vk, \sigma; m) \in \mathcal{R}_{\mathsf{PoPS}}$ *into a proof* π' *for* $(vk, \sigma'; m) \in \mathcal{R}_{\mathsf{PoPS}}$*, where* $\sigma' \leftarrow \mathsf{Rand}(\sigma)$ *is a rerandomized signature, then we call the PoP scheme* rerandomizable*.*

Definition 8 (Rerandomizability)**.** *Let* Rerand *be an efficient rerandomization procedure that, given a PoP proof* π *for* $(vk; \sigma, m) \in \mathcal{R}_{\mathsf{PoPS}}$*, outputs a proof* π' *for* $(vk; \sigma, m) \in \mathcal{R}_{\mathsf{PoPS}}$*. We say the PoP is* rerandomizable *if* π' *is always accepting whenever* π *is accepting, and the rerandomized proof is indistinguishable from a fresh proof on the rerandomized signature: for any PPT distinguisher* $\mathcal{D}$*,*

$$\left|\Pr\left[\mathcal{D}(\pi') = 1\right] - \Pr\left[\mathcal{D}(\widehat{\pi}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}, vk, (\sigma, m))$*,* $\pi' \leftarrow \mathsf{Rerand}(\pi)$*, and* $\widehat{\pi} \leftarrow \Pi.\mathsf{Prove}(\mathsf{pp}, vk, (\sigma, m))$*.*

Efficiency challenge. A naïve rerandomizable PoP instantiates the NIZK with a concrete IVC construction such as SUPERHALO. The prover first generates a proof for $\mathcal{R}_{\mathsf{PoPS}}$ and then iteratively updates it by proving the rerandomization step $\sigma' \leftarrow F_{\mathsf{re}}(\sigma)$. However, verification algorithms for legacy signatures involve non-native group operations and pairings, which are prohibitively expensive to emulate inside arithmetic circuits of general-purpose proof systems.

New construction from proof compositions. We adopt the proof-composition approach [13], [14]. This moves the expensive non-native operations (in particular, pairings) outside the circuit and proves their correctness separately using a Σ-protocol [11] based on ElGamal commitments [14].

We first recap the BLS algorithm [26] for signing a message $m \in \{0,1\}^*$ with secret key $sk \in \mathbb{F}$ and public key $vk = g_2^{sk} \in \mathbb{G}_2$, where g_2 is the generator of $\mathbb{G}_2$. The signature is $\sigma = c^{sk} \in \mathbb{G}_1$, where $c = \mathsf{Hash}(m)$ and Hash maps m into $\mathbb{G}_1$. Verification checks $e(\sigma, g_2) = e(c, vk)$. We describe the rerandomizable PoP for BLS signatures in Construction 2, following the proof composition approach of Woo *et al.* [14]. We additionally introduce a technique to reduce the recursive circuit size: instead of re-randomizing the ElGamal commitment C_c by computing $C_c' = C_c^r$ in the circuit, the prover computes $C_c' = C_c \cdot H^{v_c'}$, where $C_c = \mathsf{Commit}(c; v_c) = c \cdot H^{v_c}$. In other words, the NIVC computes the following incremental computation:

$$F^{(0)}(m) = H(m) \ \wedge \ F^{(n)}(m) = H(m) \cdot H^{v_c^{(1)}} \cdots H^{v_c^{(n)}}.$$

Construction 2 (Rerandomizable PoP for BLS Signatures)**.** *Given a BLS signature* $\sigma \coloneqq c^{sk} \in \mathbb{G}_1$*, where* $c = \mathsf{Hash}(m)$*. Assume a* Σ*-proof system* Σ *and an NIZK proof system* Π*. The prover generates a proof of possession for the BLS signature as a composition* $(C_\sigma, C_c, \pi_\Sigma, \pi_\Pi)$*. Specifically,* π_Σ *is generated by* $\Sigma.\mathsf{Prove}$ *for the relation* $\mathcal{R}_\Sigma$*,*

$$\mathcal{R}_\Sigma = \left\{ (vk, C_\sigma, C_c; \sigma, c) \colon \begin{array}{c} C_\sigma \coloneqq \mathsf{Commit}(\sigma; v_\sigma) \\ \wedge \ C_c \coloneqq \mathsf{Commit}(c; v_c) \\ \wedge \ e(\sigma, g_2) = e(c, vk) \end{array} \right\},$$

where $vk \in \mathbb{G}_2$ *and* g_2 *is the public generator of* $\mathbb{G}_2$*. Here* C_σ, C_c *are the group parts of ElGamal commitments (Definition 12) to* σ *and* c*, respectively. Moreover,* π_Π *is output by* $\Pi.\mathsf{Prove}$ *to convince that* $c = \mathsf{Hash}(m)$ *and that* c *is the same as the one committed in* C_c *(link proof). To enable rerandomization, the* Rerand *algorithm first computes the incremental step on the NIZK proof* π_Π *with the following function* F_{re}*:*

$$F_{\mathsf{re}}(C_c, v_c') = C_c \cdot H^{v_c'} = \mathsf{Commit}(c; v_c'). \tag{1}$$

The algorithm can efficiently update the Sigma proof elements accordingly. As a result, the prover obtains the composed proof of possession $(C_\sigma', C_c', \pi_\Sigma', \pi_\Pi')$ *as the rerandomized PoP for the same BLS signature* (vk, σ, c)*. Appendix C presents the detailed construction and the proof of Lemma 1.*

Lemma 1. *Construction 2 provides completeness, adaptive knowledge soundness, and zero-knowledge.*

5. Construction

In this section we provide the instantiation of SHIFT with the BLS signature scheme.

5.1. NIVC Function Definitions

We first introduce the required predicates for proving BLS-signature-based credentials. The function family handled by SUPERHALO is

$$\mathcal{F} = \{F_{\mathsf{pop}}, F_{\mathsf{re}}\} \cup \{F_\phi \colon \phi \in L_\Phi\}.$$

The transition F_{pop} starts the NIVC chain by binding the credential digest to the PoP component. The transition F_{re} rerandomizes the presentation state. For each supported predicate $\phi \in L_\Phi$, the transition F_ϕ proves $\phi(\vec{a}) = 1$.

- F_{PoP}: As described in Section 4.2, this function takes the attribute vector $\vec{a}$ as input and outputs a digest $c \coloneqq \mathsf{Hash}(\vec{a}) \in \mathbb{G}_1$ together with the corresponding ElGamal commitment.
- F_{re}: This function takes the committed digest C_c and an auxiliary (private) group element $R \in \mathbb{G}_1$ as input and outputs a rerandomized committed digest $C'_c \coloneqq C_c \cdot R$, as described in Equation 1.
- F_{ϕ_i}: Each such function takes the attribute values $\vec{a}$ as input and asserts that the predicate $\phi_i(\vec{a}) = 1$.

5.2. Full Construction

SHIFT with BLS signatures uses four building blocks: the BLS signature scheme Sig, the rerandomizable proof-of-possession construction from Construction 2, the ElGamal commitment scheme Ψ, and the SUPERHALO NIVC scheme Π. In Fig. 9, we use the RPoP construction through the following three interfaces.

- $\mathsf{RPoP.Prove}(\mathsf{vk}, \sigma, c, r, \mathsf{ctx})$ generates the composed PoP proof for a BLS witness satisfying $e(\sigma, g_2) = e(c, \mathsf{vk})$ and for its first rerandomization (σ^r, c^r). It outputs the Σ proof π_Σ, commitments C_σ, C_c, and the auxiliary group element R used by the NIVC rerandomization step.
- $\mathsf{RPoP.Rerand}(\mathsf{vk}, \sigma, c, r, z, \mathsf{ctx})$ takes the original BLS witness (σ, c), a nonzero rerandomization scalar r, and the current public digest state z as inputs, and generates a fresh non-interactive Σ proof for the rerandomized pair (σ^r, c^r). It outputs π_Σ, commitments C_σ, C_c, and the auxiliary group element $R = c^r/z$.
- $\mathsf{RPoP.Verify}(\mathsf{pp}_1, \mathsf{vk}, C_\sigma, C_c, \pi_\Sigma, \mathsf{ctx})$ verifies the non-interactive Σ transcript, including the BLS pairing equation over the committed values and the Fiat–Shamir challenge bound to the session context.

We summarize the complete protocol specification in Fig. 9.

5.3. Security Analysis

We prove that the construction in Fig. 9 satisfies the security notions in Section 3. Let Ω denote the resulting reusable anonymous credential scheme. Because Π (SUPERHALO) and the non-interactive RPoP component use Fiat–Shamir, the analysis is in the random oracle model (ROM). We give proof sketches here and defer full proofs to Appendix D. The analysis relies on the following assumptions.

- **Signature security.** Sig is EUF-CMA secure [26]. For hidden issued BLS credentials, we rely on the following assumption. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be asymmetric bilinear groups of prime order p, let g_2 generate $\mathbb{G}_2$, sample $x \leftarrow\!\$\ \mathbb{F}_p$ and $h \leftarrow\!\$\ \mathbb{G}_1 \setminus \{1\}$, and give $\mathcal{A}$ the tuple $(g_2, g_2^x, h, \{(h^{r_i}, h^{xr_i})\}_{i=1}^q)$ for independent $r_i \leftarrow\!\$\ \mathbb{F}_p^*$ and polynomial q. The *q-rerandomized-target CDH* (q-RT-CDH) assumption states that these rerandomized pairs do not enable recovery of the base value h^x.
- **RPoP security.** We use the RPoP construction from Construction 2. Its Σ component proves the BLS pairing equation over ElGamal commitments, while its NIVC/link component proves $c = \mathsf{Hash}(\vec{a})$ and rerandomization consistency. We assume the composed RPoP proof is complete, zero-knowledge, and adaptive knowledge sound in the ROM. The proof-of-exponentiation subprotocol enforces consistency between the Σ-side rerandomized commitments and the public NIVC transition.
- **NIVC security.** The function-hiding NIVC scheme Π satisfies completeness, knowledge soundness, zero-knowledge, and function-hiding; *cf.* Lemma 3.
- **Commitment hiding.** The commitment scheme Ψ is computationally hiding.

Remark 4 (q-RT-CDH and co-CDH)**.** *The q-RT-CDH assumption is equivalent to the co-CDH assumption in the generic group model [27]. Each auxiliary pair (h^{r_i}, h^{xr_i}) in the q-RT-CDH input is a co-DH tuple with base h^{r_i}. Since the scalars r_i are uniformly random and independent, the q auxiliary tuples are independent random co-DH instances. In the generic group model, an adversary making at most t group-operation queries cannot solve co-CDH with advantage better than $O(t^2/p)$, and the additional q auxiliary tuples provide no further structural advantage. Hence q-RT-CDH reduces to co-CDH in the generic group model, with a loss factor of at most q.*

Theorem 1 (Correctness)**.** *Let Ω be the construction in Fig. 9. Assume that (i) Sig is correct, (ii) Π is complete, (iii) the RPoP construction is complete, and (iv) Ψ is correct. Then for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr\left[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda) = 1\right] \le \mathsf{negl}(\lambda)$.*

Proof sketch. Conditioned on the experiment not aborting, all requested predicates are supported and satisfied, and each reuse command is applied to a well-defined state. By correctness/completeness of Sig, Π, the RPoP construction, and Ψ, each call to Present/Update/Rerand yields a token whose sub-proofs verify and whose commitments are well-formed; hence Verify accepts except with negligible probability. A union bound over polynomially many schedule steps concludes the claim. □

Theorem 2 (Unforgeability)**.** *Let Ω be the construction in Fig. 9. Assume that (i) Sig is EUF-CMA secure and BLS*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$:

1 : Run $\mathsf{pp}_0 \leftarrow\!\$\ \mathsf{Sig.Setup}(1^\lambda)$;
2 : Run $\mathsf{pp}_1 \leftarrow\!\$\ \Pi_\Sigma.\mathsf{Setup}(1^\lambda)$;
3 : Run $\mathsf{pp}_2 \leftarrow\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$;
4 : Run $\mathsf{pp}_3 \leftarrow\!\$\ \Psi.\mathsf{Setup}(1^\lambda)$;
5 : Determine the attribute space $\vec{\mathbb{A}}$;
6 : Output $\mathsf{pp} \coloneqq (\mathsf{pp}_0, \mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_3, \vec{\mathbb{A}})$.

$\mathsf{IKeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$:

1 : Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\$\ \mathsf{Sig.KeyGen}(\mathsf{pp}_0, 1^\lambda)$;
2 : Output $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{HKeyGen}(\mathsf{pp}, L_\phi) \to (\mathsf{pk}_\Pi, \mathsf{vk}_\Pi)$:

1 : Construct F_{pop} and F_{re} from Construction 2;
2 : Construct F_ϕ for each $\phi \in L_\phi$;
3 : Let $\vec{F} \coloneqq (F_{\mathsf{pop}}, F_{\mathsf{re}}, (F_\phi)_{\phi \in L_\phi})$;
4 : Run $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi) \leftarrow \Pi.\mathsf{KeyGen}(\mathsf{pp}_2, \vec{F})$;
5 : Output $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi)$.

$\mathsf{Issue}(\mathsf{pp}, \vec{a}, \mathsf{sk}) \to \mathsf{cred}$:

1 : Assert $\vec{a} \in \mathsf{pp}.\vec{\mathbb{A}}$;
2 : Run $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, \vec{a})$ (*i.e.*, $\sigma \coloneqq \mathsf{Hash}(\vec{a})^{\mathsf{sk}}$);
3 : Output $\mathsf{cred} \coloneqq (\vec{a}, \sigma)$.

$\mathsf{Present}(\mathsf{pp}, \mathsf{pk}_\Pi, \mathsf{vk}, \mathsf{cred}, \phi, \mathsf{ctx}) \to (\mathsf{pt}, S)$:

1 : Calculate $c \coloneqq \mathsf{Hash}(\mathsf{cred}.\vec{a})$;
2 : Sample $r \leftarrow\!\$\ \mathbb{F}_p^*$;
3 : Run $(\pi_\Sigma, C_\sigma, C_c, R) \leftarrow \mathsf{RPoP.Prove}(\mathsf{vk}, \mathsf{cred}.\sigma, c, r, \mathsf{ctx})$;
4 : Let $z_0 \coloneqq \perp$ and $\pi_0 \coloneqq \perp$;
5 : Let $\mathsf{w}_0 \coloneqq \mathsf{cred}.\vec{a}$ and $z_1 \coloneqq c$;
6 : $\pi_1 \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_1), (0, \pi_0, \mathsf{w}_0, \mathsf{pop}))$;
7 : Let $z_2 \coloneqq z_1$ and $\mathsf{w}_1 \coloneqq \mathsf{cred}.\vec{a}$;
8 : $\pi_2 \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_2), (1, \pi_1, \mathsf{w}_1, \phi))$;
9 : Let $z_3 \coloneqq z_2^r$;
10 : Let w_2 contain R;
11 : $\pi_3 \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_3), (2, \pi_2, \mathsf{w}_2, \mathsf{re}))$;
12 : $(\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}.\mathcal{P}(\mathsf{pp}_2, \pi_3.\vec{\mathsf{X}}, \pi_3.\vec{\mathsf{W}})$;
13 : Let $\pi_\Pi \coloneqq (\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}})$;
14 : Let $\mathsf{pt} \coloneqq (z_3, \pi_3.\vec{\mathsf{X}}, \pi_\Pi, \pi_\Sigma, C_c, C_\sigma)$;
15 : Let $S \coloneqq (\pi_3, r, \mathsf{cred}, \mathsf{vk}, z_3, \pi_\Sigma, C_c, C_\sigma)$;
16 : Output (pt, S).

$\mathsf{Update}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \phi, \mathsf{ctx}) \to (\mathsf{pt}, S')$:

1 : Parse S to obtain $(\pi_i, r, \mathsf{cred}, \mathsf{vk}, z_i, \pi_\Sigma, C_c, C_\sigma)$;
2 : Let $z_0 \coloneqq \perp$;
3 : Calculate $c \coloneqq \mathsf{Hash}(\mathsf{cred}.\vec{a})$;
4 : Let $\mathsf{w}_i \coloneqq \mathsf{cred}.\vec{a}$;
5 : Let $z_{i+1} \coloneqq z_i$;
6 : $\pi_{i+1} \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_{i+1}), (i, \pi_i, \mathsf{w}_i, \phi))$;
7 : Sample $r' \leftarrow\!\$\ \mathbb{F}_p^*$;
8 : Let $\rho \coloneqq r \cdot r'$;
9 : Run $(\pi'_\Sigma, C'_\sigma, C'_c, R) \leftarrow \mathsf{RPoP.Rerand}(\mathsf{vk}, \mathsf{cred}.\sigma, c, \rho, z_i, \mathsf{ctx})$;
10 : Let $z_{i+2} \coloneqq z_i^{r'}$;
11 : Let w_{i+1} contain R;
12 : $\pi_{i+2} \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_{i+2}), (i+1, \pi_{i+1}, \mathsf{w}_{i+1}, \mathsf{re}))$;
13 : $(\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}.\mathcal{P}(\mathsf{pp}_2, \pi_{i+2}.\vec{\mathsf{X}}, \pi_{i+2}.\vec{\mathsf{W}})$;
14 : Let $\pi_\Pi \coloneqq (\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}})$;
15 : Let $\mathsf{pt} \coloneqq (z_{i+2}, \pi_{i+2}.\vec{\mathsf{X}}, \pi_\Pi, \pi'_\Sigma, C'_c, C'_\sigma)$;
16 : Let $S' \coloneqq (\pi_{i+2}, \rho, \mathsf{cred}, \mathsf{vk}, z_{i+2}, \pi'_\Sigma, C'_c, C'_\sigma)$;
17 : Output (pt, S').

$\mathsf{Rerand}(\mathsf{pp}, \mathsf{pk}_\Pi, S, \mathsf{ctx}) \to (\mathsf{pt}, S')$:

1 : Parse S to obtain $(\pi_i, r, \mathsf{cred}, \mathsf{vk}, z_i, \pi_\Sigma, C_c, C_\sigma)$;
2 : Let $z_0 \coloneqq \perp$;
3 : Calculate $c \coloneqq \mathsf{Hash}(\mathsf{cred}.\vec{a})$;
4 : Sample $r' \leftarrow\!\$\ \mathbb{F}_p^*$;
5 : Let $\rho \coloneqq r \cdot r'$;
6 : Run $(\pi'_\Sigma, C'_\sigma, C'_c, R) \leftarrow \mathsf{RPoP.Rerand}(\mathsf{vk}, \mathsf{cred}.\sigma, c, \rho, z_i, \mathsf{ctx})$;
7 : Let $z_{i+1} \coloneqq z_i^{r'}$;
8 : Let w_i contain R;
9 : $\pi_{i+1} \leftarrow \Pi.\mathsf{Prove}(\mathsf{pk}_\Pi, (z_0, z_{i+1}), (i, \pi_i, \mathsf{w}_i, \mathsf{re}))$;
10 : $(\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}.\mathcal{P}(\mathsf{pp}_2, \pi_{i+1}.\vec{\mathsf{X}}, \pi_{i+1}.\vec{\mathsf{W}})$;
11 : Let $\pi_\Pi \coloneqq (\mathsf{x}, \mathsf{w}, \pi_{\mathsf{agg}})$;
12 : Let $\mathsf{pt} \coloneqq (z_{i+1}, \pi_{i+1}.\vec{\mathsf{X}}, \pi_\Pi, \pi'_\Sigma, C'_c, C'_\sigma)$;
13 : Let $S' \coloneqq (\pi_{i+1}, \rho, \mathsf{cred}, \mathsf{vk}, z_{i+1}, \pi'_\Sigma, C'_c, C'_\sigma)$;
14 : Output (pt, S').

$\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}, \phi_1, \ldots, \phi_n, \mathsf{ctx}) \to b$:

1 : Parse pt to obtain $(z_i, \vec{\mathsf{X}}, \pi_\Pi, \pi_\Sigma, C_c, C_\sigma)$;
2 : $b_0 \leftarrow \mathsf{Aggregate}.\mathcal{V}(\mathsf{pp}_2, \vec{\mathsf{X}}, \pi_\Pi.\pi_{\mathsf{agg}})$;
3 : // $\mathsf{Aggregate}.\mathcal{V}$ checks all ℓ accumulator slots (Construction 1).
4 : Parse vk_Π into $(\mathsf{vk}_{\mathsf{pop}}, \mathsf{vk}_{\mathsf{re}}, (\mathsf{vk}_\psi)_{\psi \in L_\phi})$;
5 : Compute $\mathsf{vk}^* \leftarrow \mathsf{vk}_{\mathsf{pop}} \oplus \mathsf{vk}_{\mathsf{re}} \oplus \bigoplus_{j \in [n]} \mathsf{vk}_{\phi_j}$;
6 : // vk^* is the KZG linear combination of selector-polynomial
7 : commitments for the $n+2$ active predicate keys; cost $O(n)$.
8 : $b_1 \leftarrow \mathsf{NIZK}_{\mathsf{plonk}}.\mathsf{Verify}(\mathsf{vk}^*, \pi_\Pi.\mathsf{x}, \pi_\Pi.\mathsf{w})$;
9 : $b_2 \leftarrow \mathsf{RPoP.Verify}(\mathsf{pp}_1, \mathsf{vk}, C_\sigma, C_c, \pi_\Sigma, \mathsf{ctx})$;
10 : Outputs $b \coloneqq b_0 \wedge b_1 \wedge b_2$.

Figure 9: The instantiation of SHIFT with BLS signatures.

hidden issued credentials satisfy q-RT-CDH for polynomial q, (ii) Π is knowledge sound, and (iii) the RPoP construction is adaptive knowledge sound. Then for every PPT adversary $\mathcal{A}$ there exist PPT reductions $\mathcal{B}_{\mathsf{Sig}}, \mathcal{B}_{\mathsf{RT\text{-}CDH}}, \mathcal{B}_{\Pi}, \mathcal{B}_{\mathsf{RPoP}}$ and a negligible function $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{uf}}_{\Omega}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{Sig}}(\mathcal{B}_{\mathsf{Sig}}) + \mathsf{Adv}^{q\text{-}\mathsf{RT\text{-}CDH}}_{\mathsf{BLS}}(\mathcal{B}_{\mathsf{RT\text{-}CDH}}) + \mathsf{Adv}^{\mathsf{ks}}_{\Pi}(\mathcal{B}_{\Pi}) + \mathsf{Adv}^{\mathsf{ks}}_{\mathsf{RPoP}}(\mathcal{B}_{\mathsf{RPoP}}) + \mathsf{negl}(\lambda)$.

Proof sketch. Suppose $\mathcal{A}$ wins $\mathsf{Exp}^{\mathsf{uf}}$ with an accepting fresh token pt^*. By knowledge soundness of Π, we extract an attribute vector $\vec{a}^*$ satisfying all claimed predicates. By adaptive knowledge soundness of RPoP construction, we extract a rerandomized witness $(\widetilde{\sigma}^*, \widetilde{c}^*, \rho^*)$ satisfying the BLS pairing equation and $\widetilde{c}^* = \mathsf{Hash}(\vec{a}^*)^{\rho^*}$ for a nonzero ρ^*. Thus $\sigma^* := (\widetilde{\sigma}^*)^{1/\rho^*}$ is a valid BLS signature on $\vec{a}^*$. By the winning condition of $\mathsf{Exp}^{\mathsf{uf}}$, no revealed credential satisfies all predicates $(\phi_1^*, \ldots, \phi_n^*)$, so $\vec{a}^*$ does not match any revealed attribute vector. Therefore, either (i) $\vec{a}^*$ was never signed by the issuer, giving an EUF-CMA forgery for Sig, or (ii) $\vec{a}^*$ belongs to an issued-but-unrevealed credential, in which case computing $\sigma^* = \mathsf{Hash}(\vec{a}^*)^{\mathsf{sk}}$ from vk and $\mathsf{Hash}(\vec{a}^*)$ after observing polynomially many rerandomized tokens $(\mathsf{Hash}(\vec{a}^*)^r, (\sigma^*)^r)$ violates q-RT-CDH. The reduction $\mathcal{B}_{\mathsf{RT\text{-}CDH}}$ simulates all oracle outputs (including ZK proofs inside holder-oracle tokens) using the ZK simulators for Π and RPoP; the auxiliary rerandomized group elements (h^{r_i}, h^{xr_i}) in the q-RT-CDH input correspond exactly to the rerandomized signature components in oracle tokens, so no information beyond the formal assumption leaks to $\mathcal{A}$. □

Lemma 2 (Self-randomization in prime-order groups)**.** *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. For any fixed $x \in \mathbb{G} \setminus \{1\}$ and uniform $r \leftarrow\$ \mathbb{F}_p$, the distribution of x^r is statistically close to uniform over $\mathbb{G}$. In particular, for any two fixed non-identity elements $x_0, x_1 \in \mathbb{G}$, the random variables x_0^r and x_1^r are identically distributed.*

Proof sketch. Exponentiation by any nonzero $r \in \mathbb{F}_p^*$ is a permutation of a prime-order group, hence x^r is uniform. Sampling $r \leftarrow\$ \mathbb{F}_p$ differs only by the event $r = 0$, which occurs with probability $1/p \in \mathsf{negl}(\lambda)$. □

Theorem 3 (Anonymity)**.** *Let Ω be the construction in Fig. 9. Assume that (i) Π is zero-knowledge and function-hiding, (ii) the RPoP construction is zero-knowledge, and (iii) Ψ is hiding. Then for every PPT adversary $\mathcal{A}$ there exist PPT reductions $\mathcal{B}^{\mathsf{zk}}_{\Pi}, \mathcal{B}^{\mathsf{fh}}_{\Pi}, \mathcal{B}_{\mathsf{RPoP}}$, and $\mathcal{B}_{\Psi}$ and a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{anon}}_{\Omega}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}^{\mathsf{zk}}_{\Pi}) + \mathsf{Adv}^{\mathsf{fh}}_{\Pi}(\mathcal{B}^{\mathsf{fh}}_{\Pi}) + \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{RPoP}}(\mathcal{B}_{\mathsf{RPoP}}) + \mathsf{Adv}^{\mathsf{hide}}_{\Psi}(\mathcal{B}_{\Psi}) + \mathsf{negl}(\lambda)$.

Theorem 4 (Unlinkability)**.** *Let Ω be the construction in Fig. 9. Assume that (i) Π is zero-knowledge and function-hiding, (ii) the RPoP construction is zero-knowledge, and (iii) Ψ is hiding. Then for every PPT adversary $\mathcal{A}$ there exist PPT reductions $\mathcal{B}^{\mathsf{zk}}_{\Pi}, \mathcal{B}^{\mathsf{fh}}_{\Pi}, \mathcal{B}_{\mathsf{RPoP}}$, and $\mathcal{B}_{\Psi}$ and a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Adv}^{\mathsf{unlink}}_{\Omega}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}^{\mathsf{zk}}_{\Pi}) + \mathsf{Adv}^{\mathsf{fh}}_{\Pi}(\mathcal{B}^{\mathsf{fh}}_{\Pi}) + \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{RPoP}}(\mathcal{B}_{\mathsf{RPoP}}) + \mathsf{Adv}^{\mathsf{hide}}_{\Psi}(\mathcal{B}_{\Psi}) + \mathsf{negl}(\lambda)$.

Proof sketch for Theorems 3 and 4. Fix any admissible challenge in either $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}$ or $\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}$. We bound the corresponding advantage via a sequence of five hybrid games over all challenge tokens generated by Run; see Appendix D for the complete proof. In brief: first, zero-knowledge of Π replaces NIVC proofs with simulated ones (preserving the real public statements); second, function-hiding of Π replaces the visible update-function trace with the ideal trace (hiding which functions from $\{F_{\mathsf{PoP}}, F_{\mathsf{re}}, F_{\phi_i}\}$ were applied); third, zero-knowledge of the RPoP construction replaces composed PoP proofs with simulated proofs against the real public statements; fourth, commitment hiding of Ψ replaces credential-dependent commitments with fixed dummy values, with the RPoP proofs simultaneously re-simulated against the new dummy statements (the re-simulation is valid because the step-three simulator does not use the credential witnesses); fifth, Lemma 2 implies that all rerandomized digest values are statistically close to uniform and independent of the underlying credential. In the final hybrid, every challenge token is generated from simulated proofs, dummy commitments, and uniform group elements, so the joint transcript no longer depends on the hidden bit. □

6. Evaluation

6.1. Setup and Implementation

All benchmarks were conducted on a MacBook Air equipped with an M1 CPU, 16 GB of RAM, and a 256 GB SSD. Each timing measurement reports the *median over* 10 *independent runs* executed sequentially on a single thread. No warm-caching was applied between runs. Standard deviations were below 5% of the median in all cases and are omitted for clarity. Reported times include both the NIVC proving step and the RPoP Σ-protocol computation.

Our system is implemented in Rust (v1.90.0-nightly). We set the security parameter to $\lambda = 128$ and use Scroll's fork of Halo2, instantiated with the KZG polynomial commitment scheme. The proof system operates over the BN254/Grumpkin cycle of curves. We evaluate two signature schemes: BLS over BN254 and ECDSA[1] over P-256. SHA-256 is used as the hash function throughout.

Woo *et al.* [14] is the state-of-the-art PoP, relying on lookup tables for circuit optimization. For fair comparison, we adapt their PoP into a full AC system by integrating predicate checks within its circuits. We also include generic zk-SNARKs (Plonk/Groth16) as baselines, which verify signatures and perform predicate checks without optimizations. We evaluate three variants of SHIFT:

- Full: Generates a fresh presentation from scratch, equivalent to a single-session cost with no prior state.
- Update: Derives a new presentation from an existing state by appending a new predicate proof, without re-proving the signature-verification relation.

1. For ECDSA, we adopt a naïve instantiation where F_{PoP} encodes the full signature-verification relation and F_{re} is a dummy circuit that outputs the F_{PoP} result unchanged. Rerandomization therefore costs 0 additional constraints for ECDSA; the per-session gain comes entirely from skipping the signature circuit on Update.

TABLE 1: Estimated proving constraints of different schemes for BLS and ECDSA.

Scheme	ECDSA	BLS
Full	$l_1 \cdot H + 2 \cdot \mathbb{G}^{\times} + 1 \cdot \mathbb{G}^{+} + \phi \approx l_1 \cdot 2^{13} + 2 \cdot 2^{18} + 2^9 + \lvert\phi\rvert$	$l_1 \cdot H + 1 \cdot \mathbb{G}_1^{\times} + 2 \cdot \mathbb{G}_1^{+} + \phi \approx l_1 \cdot 2^{13} + 2^{18} + 2 \cdot 2^9 + \lvert\phi\rvert$
Update	$\phi \approx \lvert\phi\rvert$	$1 \cdot \mathbb{G}_1^{+} + \phi \approx 2^9 + \lvert\phi\rvert$
Rerand	0	$1 \cdot \mathbb{G}_1^{+} \approx 2^9$
Woo *et al.*	$l_1 \cdot H + 1 \cdot \mathbb{G}^{\times} + 1 \cdot \mathbb{G}^{+} + \phi \approx l_1 \cdot 2^{13} + 2^{18} + 2^9 + \lvert\phi\rvert$	–
Plonk/Groth16	$l_1 \cdot H + 2 \cdot \mathbb{G}^{\times} + 1 \cdot \mathbb{G}^{+} + \phi \approx l_1 \cdot 2^{15} + 2 \cdot 2^{20} + 2^{12} + \lvert\phi\rvert$	$l_1 \cdot H + 2P + \phi \approx l_1 \cdot 2^{15} + 2 \cdot 2^{22} + \lvert\phi\rvert$

[1] l_1 represents the length of signed message (in 512-bit chunks) and H denotes the hash operation over a 512-bit chunk.
[2] ϕ denotes the predicate check in circuit with $\lvert\phi\rvert$ constraints.
[3] P denotes the pairing operation. $\mathbb{G}_1^{+}, \mathbb{G}^{+}$ denote point multiplication operations. $\mathbb{G}_1^{\times}$ denotes point exponentiation operations.
[4] Plonk/Groth16 refers to directly proving signature verification without optimizations such as lookup tables.
[5] BLS is not in the scope of Woo *et al.*

TABLE 2: Estimated proof sizes and verification costs of different schemes for various signature algorithms.

Scheme	Signatures	Presentation Size	Verification Cost
SHIFT	ECDSA	$7\lvert\mathbb{G}_1\rvert + 7\lvert\mathbb{F}\rvert$	$2P + 18\mathbb{G}_1$
SHIFT	BLS	$7\lvert\mathbb{G}_1\rvert + 7\lvert\mathbb{F}\rvert + 1\mathbb{G}_t'$ $+2\mathbb{F}' + 8\mathbb{G}_1'$	$2P + 18\mathbb{G}_1$ $+1\mathbb{G}_t' + 2P'$
Woo *et al.*	ECDSA	$3\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$ $+4\lvert\mathbb{G}'\rvert + 4\lvert\mathbb{F}'\rvert$	$4P + l\mathbb{G}_1$ $+10\mathbb{G}'$
Plonk	ECDSA, BLS	$7\lvert\mathbb{G}_1\rvert + 7\lvert\mathbb{F}\rvert$	$2P + 18\mathbb{G}_1$
Groth16	ECDSA, BLS	$2\lvert\mathbb{G}_1\rvert + 1\lvert\mathbb{G}_2\rvert$	$3P + l\mathbb{G}_1$

[1] $\lvert\cdot\rvert$ means the length of an element in a group or field and l denotes the number of public inputs.
[2] $P, \mathbb{G}_1, \mathbb{G}_2$ denote the proof system's pairing operation, group point exponentiation operations, respectively. Signatures' underlying curves' operations or elements are marked with a superscript like $\mathbb{G}_T', \mathbb{G}', P', \mathbb{F}'$.

- Rerand: Derives a fresh, unlinkable presentation from an existing state with no new predicate change.

We do not include HyperNova [17] as a direct timing baseline because HyperNova targets general-purpose IVC without the proof-composition optimization that moves pairing computations outside the circuit. A fair end-to-end timing comparison would require re-implementing a HyperNova-based AC with the same PoP composition technique and then adding the trace-hiding machinery needed for our flow-mode security goal. Accordingly, we do not claim a timing advantage over such an optimized HyperNova-based AC implementation; our comparison is against the implemented PoP and SNARK baselines and against the standard public-state IVC interface discussed in Section 7.

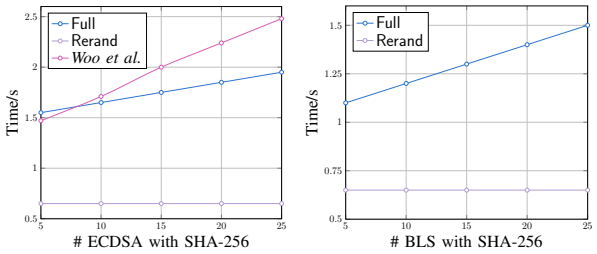
6.2. Theoretical Analysis

Proving cost. Table 1 reports estimated circuit constraints, which serve as a primary metric for proving complexity. For fresh presentation (Full), the dominant costs are message hashing, signature-related curve operations, and predicate checking. For ECDSA, Full is in the same asymptotic range as Woo *et al.*, with one additional point exponentiation; for BLS, it avoids the in-circuit pairings required by generic zk-SNARK baselines. For the frequent stateful operations, SHIFT has lower estimated proving cost in this model: Update reduces the ECDSA proving cost to the predicate circuit alone ($\lvert\phi\rvert$), and reduces the BLS cost to one $\mathbb{G}_1$ multiplication plus the predicate ($\approx 2^9 + \lvert\phi\rvert$). Rerand removes predicate re-evaluation entirely, yielding zero additional ECDSA circuit constraints and only one $\mathbb{G}_1$ multiplication for BLS ($\approx 2^9$). By contrast, Woo *et al.* and generic zk-SNARKs require full re-proving of the signature and all predicates for every session. *Communication and verification.* Table Table 2 reports proof sizes and verification costs. SHIFT produces compact constant-size presentations: the ECDSA token contains $7\lvert\mathbb{G}_1\rvert + 7\lvert\mathbb{F}\rvert$, while the BLS token additionally carries signature-curve elements ($1\mathbb{G}_t' + 2\mathbb{F}' + 8\mathbb{G}_1'$). Verification cost is $O(1)$ in the number of predicates, requiring a fixed $2P + 18\mathbb{G}_1$ for ECDSA plus $1\mathbb{G}_t' + 2P'$ for BLS. This removes the l-dependent group-operation term present in Woo *et al.*'s verification cost. Overall, SHIFT improves prover efficiency for stateful presentations while retaining compact proofs and lightweight verification.

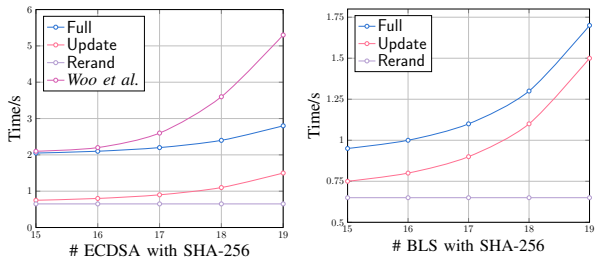
6.3. Experimental Results

Credential presentation time is primarily influenced by message length and predicate complexity. To align with the SHA-256 chunk size, we set attribute lengths as multiples of 512 bits. Predicates are instantiated as arithmetic range-check circuits: each predicate asserts that a single 64-bit attribute lies within a specified range, and the circuit is padded with no-op constraints to reach the target constraint count. In the predicate-size experiments, we vary the predicate circuit size from 2^{15} to 2^{19} constraints while fixing the message size to 10×512 bits. Woo *et al.* does not support BLS signatures and is included only for ECDSA. We omit naïve Plonk/Groth16 baselines from the figures because their proving times without optimization exceed tens of seconds for a bare PoP, making the plots unreadable at our scale.

Impact of message size. Fig. 10a reports core proving costs without predicates, as the message size varies from 5×512 to 25×512 bits. Full grows modestly with message size as hashing more chunks adds constraints: from 1.55 s to 1.95 s for ECDSA and from 1.1 s to 1.5 s for BLS. Woo *et al.* grows from 1.47 s to 2.48 s over the same range, making Full comparable for fresh sessions. The primary gain is on



(a) Presentation cost vs. signed message size (multiples of 512 bits, no predicate checks).



(b) Presentation cost vs. predicate size (constraints in $\log_2$ scaling, message fixed at 10×512 bits).

Figure 10: Experimental results for SHIFT: Points show medians over 10 runs; standard deviations are below 5%.

the stateful path: Rerand achieves constant performance ($\approx$ 0.65 s) for both schemes, independent of attribute length.

Impact of predicate size. Fig. 10b evaluates predicate complexity from 2^{15} to 2^{19} constraints with a fixed message size. Woo *et al.* rises sharply from 2.1 s to 5.3 s. Full reaches 2.8 s for ECDSA and 1.7 s for BLS at the largest predicate. The stateful Update path reduces ECDSA cost to 1.5 s at the largest predicate, a $3.5\times$ improvement over Woo *et al.* (5.3 s vs. 1.5 s) for incremental predicate updates. Rerand remains at ≈ 0.65 s regardless of predicate size.

Verification cost. Proof sizes for all schemes do not exceed 2 KB, and verification times are below 50 ms. Both SHIFT and Woo *et al.* use zk-SNARKs for the circuit component, so the proof size and circuit verification cost are fixed. The Σ-protocol component outside the circuit contributes only a small constant overhead.

7. Related Work

Camenisch and Lysyanskaya [28] established the foundation of anonymous credentials, evolving through P-signatures [29], threshold issuance like Coconut [30], and randomizable credentials using SPS-EQ [7]. While modern BBS-based schemes achieve efficient presentations for algebraic predicates via signature rerandomization, their supported predicate class remains narrow, they lack interoperability with deployed ECDSA or BLS signatures, and presentations are inherently stateless. Concurrently, systems like Cinderella [31], UCBlocker [8], CanDID [32], and zk-creds [1] apply zero-knowledge proofs to identity and verifiable credentials. SHIFT is complementary to this line of work: it targets holders of legacy signatures requiring repeated, stateful presentations under incrementally evolving policies, formalizing a security framework that specifically captures intra-flow linkability.

Recursive SNARKs for IVC have advanced rapidly, from accumulation in Halo [22] to folding schemes like Nova [15] and its multi-function generalizations, SuperNova [16] and KiloNova [24]. HyperNova [17] further extends this framework to customizable constraint systems (CCS) while achieving zero-knowledge. The standard public-state interfaces of SuperNova-style multi-function IVCs, including HyperNova-style deployments, maintain per-function accumulator state and update the active component in a way that can expose the applied transition when consecutive proof states are compared. This structural trace is harmless for many IVC applications, but it is exactly the information that flow-mode unlinkability must hide in reusable ACs. General-purpose IVC also does not by itself provide the proof-composition interface used here to keep legacy-signature algebra outside the recursive circuit. SUPERHALO addresses this application-specific gap by adding uniform dummy updates on all ℓ slots, so that the public update pattern no longer identifies the active transition.

SigmaBus [13] pioneered the proof-composition paradigm to efficiently prove signature possession by binding a Σ-protocol to an arithmetic circuit. Woo *et al.* [14] successfully realized this for ECDSA and RSA signatures, substantially reducing single-session presentation overhead. SHIFT builds on the proof-composition approach of Woo *et al.* and adds two mechanisms needed for the stateful regime: (i) an IVC-driven rerandomization mechanism that updates the Fiat–Shamir challenge without the prohibitive cost of re-proving the full signature-verification relation, and (ii) integration with SUPERHALO so that the public update pattern does not reveal the rerandomization trace to the verifier.

8. Conclusion

We presented SHIFT, a reusable anonymous credential framework for stateful, unlinkable presentations. The framework formalizes a two-mode unlinkability notion, including *flow-mode unlinkability* for tokens derived from the same reuse chain. Its stateful presentation model amortizes the cost of proving possession of legacy signatures by supporting incremental predicate updates and rerandomization.

SHIFT combines SUPERHALO, a function-hiding non-uniform IVC scheme, with a rerandomizable proof-of-possession mechanism. Function hiding is enforced through uniform dummy updates across all ℓ accumulator slots, while the PoP component avoids embedding BLS signature verification inside the recursive circuit. Our prototype shows that this design is practical on commodity hardware: rerandomization takes 0.65 s, verification is below 50 ms, and proof sizes remain under 2 KB. These results indicate reusable anonymous authentication at speed.

References

[1] M. Rosenberg, J. D. White, C. Garman, and I. Miers, "zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 790–808.

[2] M. Rosenberg, M. Maller, and I. Miers, "Snarkblock: Federated anonymous blocklisting from hidden common input aggregate proofs," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 948–965.

[3] A. Naveh and E. Tromer, "Photoproof: Cryptographic image authentication for any set of permissible transformations," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 255–271.

[4] M. H. A. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Systems Journal*, vol. 7, no. 2, pp. 249–261, 2013.

[5] M. K. Franklin and H. Zhang, "Unique ring signatures: A practical construction," in *Financial Cryptography and Data Security*, 2013, pp. 162–170.

[6] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Topics in Cryptology – CT-RSA*, 2016, pp. 111–126.

[7] G. Fuchsbauer, C. Hanser, and D. Slamanig, "Structure-preserving signatures on equivalence classes and constant-size anonymous credentials," *J. Cryptol.*, vol. 32, no. 2, pp. 498–546, 2019.

[8] C. Du, H. Yu, Y. Xiao, Y. T. Hou, A. D. Keromytis, and W. Lou, "UCBlocker: Unwanted call blocking using anonymous authentication," in *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, 2023, pp. 445–462.

[9] R. Shi, H. Feng, Y. Yang, F. Yuan, Y. Li, H. Pang, and R. H. Deng, "Threshold attribute-based credentials with redactable signature," *IEEE Trans. Serv. Comput.*, vol. 16, no. 5, pp. 3751–3765, 2023.

[10] S. Xie, R. Shi, Y. Yang, H. Xie, Y. Li, and R. H. Deng, "Acccred: Improved accountable anonymous credentials with dynamic triple-hiding committees," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 3, pp. 1961–1975, 2025.

[11] T. Attema and R. Cramer, "Compressed ς-protocol theory and practical application to plug & play secure algorithmics," in *Advances in Cryptology – CRYPTO 2020*, vol. 12172. Springer, 2020, pp. 513–543.

[12] K. Wang, J. Gao, Q. Wang, J. Zhang, Y. Li, Z. Guan, and Z. Chen, "Hades: Practical decentralized identity with full accountability and fine-grained sybil-resistance," in *Annual Computer Security Applications Conference, ACSAC 2023, Austin, TX, USA, December 4-8, 2023*. ACM, 2023, pp. 216–228.

[13] G. Kadianakis, M. Maller, and A. Novakovic, "Sigmabus: Binding Sigmas in circuits for fast curve operations," IACR Cryptol. ePrint Arch. 2023/1406, 2023. [Online]. Available: https://eprint.iacr.org/2023/1406

[14] A. P. Y. Woo, A. Ozdemir, C. Sharp, T. Pornin, and P. Grubbs, "Efficient proofs of possession for legacy signatures," in *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*. IEEE, 2025, pp. 3291–3308.

[15] A. Kothapalli, S. T. V. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.

[16] A. Kothapalli and S. T. V. Setty, "SuperNova: Proving universal machine executions without universal circuits," IACR Cryptol. ePrint Arch. 2022/1758, 2022. [Online]. Available: https://eprint.iacr.org/2022/1758

[17] ——, "Hypernova: Recursive arguments for customizable constraint systems," in *Advances in Cryptology – CRYPTO 2024, Part V*, ser. Lecture Notes in Computer Science, vol. 14924. Springer, 2024, pp. 345–378.

[18] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International conference on the theory and application of cryptology and information security*, vol. 6477. Springer, 2010, pp. 177–194.

[19] J. Kim, A. Kothapalli, O. Chardouvelis, R. S. Wahby, and P. Grubbs, "ALPACA: anonymous blocklisting with constant-sized updatable proofs," in *IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025*. IEEE, 2025, pp. 3364–3382.

[20] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, "Issuer-hiding attribute-based credentials," IACR Cryptol. ePrint Arch. 2022/213, 2022. [Online]. Available: https://eprint.iacr.org/2022/213

[21] P. Valiant, "Incrementally verifiable computation or proofs of knowledge imply time/space efficiency," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, vol. 4948. Springer, 2008, pp. 1–18.

[22] S. Bowe, J. Grigg, and D. Hopwood, "Halo: Recursive proof composition without a trusted setup," IACR Cryptol. ePrint Arch. 2019/1021, 2019. [Online]. Available: https://eprint.iacr.org/2019/1021

[23] D. Boneh, J. Drake, B. Fisch, and A. Gabizon, "Halo infinite: Proof-carrying data from additive polynomial commitments," in *Annual International Cryptology Conference*, vol. 12825. Springer, 2021, pp. 649–680.

[24] T. Zheng, S. Gao, Y. Guo, and B. Xiao, "KiloNova: Preprocessing folding-based SNARKs for machine executions," IACR Cryptol. ePrint Arch. 2023/1579, 2023. [Online]. Available: https://eprint.iacr.org/2023/1579

[25] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," IACR Cryptol. ePrint Arch 2019/953, 2019.

[26] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.

[27] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1997, pp. 256–266.

[28] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2001, pp. 93–118.

[29] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "P-signatures and noninteractive anonymous credentials," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, vol. 4948. Springer, 2008, pp. 356–374.

[30] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[31] A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, and B. Parno, "Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation," in *2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 235–254.

[32] D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller, "Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 1348–1366.

[33] B. Bünz, A. Chiesa, W. Lin, P. Mishra, and N. Spooner, "Proof-carrying data without succinct arguments," in *Annual International Cryptology Conference*. Springer, 2021, pp. 681–710.

[34] B. Libert, "Simulation-extractable KZG polynomial commitments and applications to HyperPlonk," in *PKC Part II*, 2024, pp. 68–98.

[35] privacy-ethereum, "A collection of elliptic curves for zkcrypto traits," https://github.com/privacy-ethereum/halo2curves.

Appendix A. Formal Definitions

Let $\mathbb{G}$ be a cyclic group of prime order p with additive notation, and let $\mathbb{F}_p$ denote its associated scalar field. We denote by $\mathbb{F}_p[X]$ the ring of polynomials with coefficients in $\mathbb{F}_p$. For any positive integer $n \in \mathbb{N}^+$, let $[n] \coloneqq \{1, \ldots, n\}$. For a set S, $|S|$ denotes its cardinality. A vector is denoted as $\vec{f} \coloneqq (f_1, \ldots, f_n) \in \mathbb{F}_p^n$, and the concatenation of two vectors $\vec{f}$ and $\vec{g}$ is $(\vec{f}, \vec{g})$. An NP relation $\mathcal{R}$ is defined over pairs of public statement x and witness w, such that $(\mathsf{x}; \mathsf{w}) \in \mathcal{R}$. We use PPT to denote probabilistic polynomial time.

Definition 9 (KZG Polynomial Commitment Scheme [18])**.** *KZG consists of the following algorithms:*

- $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, D)$*: Sample a bilinear group tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, g_1, g_2, e)$*, a secret* $\beta \in \mathbb{F}_p$*, and set* $\vec{g}_1 \coloneqq (g_1, \beta g_1, \ldots, \beta^D g_1) \in \mathbb{G}_1^{D+1}$*. Output* $\mathsf{pk} \coloneqq ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, p, g_1, g_2, e), \vec{g}_1)$ *and* $\mathsf{vk} \coloneqq (g_1, g_2, \beta g_2)$*.*
- $C \leftarrow \mathsf{Commit}(\mathsf{pk}, f)$*: For* $f(X) = \sum_{j=0}^{d} f_j X^j \in \mathbb{F}_p[X]$ *with* $d \leq D$*, output* $C \coloneqq \sum_{j=0}^{d} f_j \cdot \beta^j g_1 \in \mathbb{G}_1$*.*
- $\pi \leftarrow \mathsf{Open}(\mathsf{pk}, f, z)$*: For* $z \in \mathbb{F}_p$*, compute* $w(X) \coloneqq \frac{f(X)-f(z)}{X-z} = \sum_{j=0}^{d-1} w_j X^j$ *and output* $\pi \coloneqq \sum_{j=0}^{d-1} w_j \cdot \beta^j g_1 \in \mathbb{G}_1$*.*
- $b \leftarrow \mathsf{Check}(\mathsf{vk}, C, z, v, \pi)$*: Output* $b = 1$ *iff* $e(C - vg_1, g_2) = e(\pi, \beta g_2 - z g_2)$*.*

Appendix B. Instantiation of SUPERHALO

B.1. Toolbox

We adopt the framework used in Halo and its variants [22], [23], [33], which integrates a committed relation, a polynomial commitment aggregation scheme, and a cycle of curves to construct an efficient recursive SNARK.

B.1.1. Plonkish relations. We first review the relations in the Plonk proof system [25]. Plonk offers better expressiveness by supporting custom gates and lookup operations than R1CS. The prover arithmetizes the circuit computation into a polynomial relation over the gate wires, reduces its validity over the subgroup $H \subseteq \mathbb{F}_p$ to evaluations at random points via an interactive protocol, and then uses polynomial commitments to obtain a committed Plonkish relation for $\mathsf{NIZK}_{\mathsf{plonk}} = (\mathsf{Prove}, \mathsf{Verify})$, whose completeness and simulation extractability follow from Libert [34].

Definition 10 (Committed Plonkish Relation)**.** *Consider a finite field* $\mathbb{F}_p$ *and a polynomial commitment scheme* PCS*. Let public parameters* pp *consist of the degree bound* $d \in \mathbb{N}$ *and commitment parameters* $\mathsf{pp}_{\mathsf{PCS}}$*. The committed Plonkish relation is an NP relation* $\mathcal{R}_{\mathsf{plonk}}$*, where a witness* $\mathsf{w} = (f_1, \ldots, f_k)$ *for a statement* $\mathsf{x} = (C_1, \ldots, C_k, z_1, \ldots, z_k, v_1, \ldots, v_k)$ *satisfies* $\mathcal{R}_{\mathsf{plonk}}$ *if (1)* $f_i(z_i) = v_i$*, (2)* $C_i = \mathsf{PCS.Commit}(\mathsf{pk}, f_i)$ *for all* $i \in [k]$*, and (3)* $\deg(f_i) \leq d$ *for all* $i \in [k]$*.*

B.1.2. PCS Aggregation. We adopt the PCS aggregation scheme [23], which is the core technique for recursive constructions. The PCS aggregation takes k PCS instance tuples $(C_1, z_1, v_1), \ldots, (C_k, z_k, v_k)$ and outputs an aggregated tuple (C^*, z^*, v^*) such that opening $C^* \in \mathbb{G}$ at $z^* \in \mathbb{F}_p$ to $v^* \in \mathbb{F}_p$ suffices to certify each C_i opened at z_i to v_i for $i \in [k]$. We formalize this scheme in Definition 11. We utilize KZG [18] as the underlying PCS and adopt an aggregation protocol of Boneh *et al.* [23].

Definition 11 (PCS Aggregation [23])**.** *Let* $\mathsf{PCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Check})$ *be a polynomial commitment scheme over* $\mathbb{G}$*. An aggregation scheme for* PCS *is a public-coin interactive protocol* $\mathsf{Aggregate}$ *with public inputs* $\vec{C} = (C_1, \ldots, C_k) \in \mathbb{G}^k$*,* $\vec{z} \in \mathbb{F}_p^k$*,* $\vec{v} \in \mathbb{F}_p^k$*, and private inputs* $\vec{f} \in \mathbb{F}_p^{(<d)}[X]^k$ *such that* $f_i(z_i) = v_i \wedge C_i = \mathsf{Commit}(f_i)$ *for all* $i \in [k]$*:*

$$((f^*; C^*, z^*, v^*), \pi_{\mathsf{agg}}) \leftarrow \mathsf{Aggregate}(\mathcal{P}(\vec{f}, \vec{z}), \mathcal{V}(\vec{C}, \vec{z}, \vec{v})),$$

where f^* *satisfies* $f^*(z^*) = v^* \wedge C^* = \mathsf{Commit}(f^*)$*, and* π_{agg} *is an aggregation proof.*

B.1.3. Cycle of curves. Building a recursive argument requires the prover to represent the verification or aggregation of the previous step's proof as a circuit and generate a proof for this circuit. This incurs significant costs for group operations in circuits. To mitigate these costs, most recursive arguments rely on a cycle of elliptic curves, *e.g.*, BN254/Grumpkin [35]. For two groups $\mathbb{G}_1, \mathbb{G}_2$ with scalar/base fields $\mathbb{F}_1, \mathbb{F}_2$ and $\mathbb{F}_2, \mathbb{F}_1$, respectively, group operations in $\mathbb{G}_1$ can be efficiently represented over the base field $\mathbb{F}_2$ (the scalar field of $\mathbb{G}_2$), and vice versa.

B.2. SUPERHALO Security

Lemma 3 (Function-hiding NIVC Security)**.** *The NIVC scheme* NIVC *in Construction 1 satisfies function-hiding. Furthermore, it satisfies completeness and knowledge soundness in the random oracle model if the underlying polynomial commitment scheme and* $\mathsf{NIZK}_{\mathsf{plonk}}$ *provide knowledge soundness. It satisfies zero-knowledge and function-hiding if the underlying* PCS *provides hiding and the underlying* $\mathsf{NIZK}_{\mathsf{plonk}}$ *provides zero-knowledge.*

Proof. Completeness follows directly from the correctness of the PCS aggregation protocol and the completeness of $\mathsf{NIZK}_{\mathsf{plonk}}$. For every honest step, the active aggregation verifies for $(\vec{\mathsf{X}}_i[\mathsf{idx}], \mathsf{x}_i)$, every passive aggregation verifies for $(\vec{\mathsf{X}}_i[j], \mathsf{x}_\perp)$, and the Plonkish proof verifies the transition relation defining z_{i+1}.

Knowledge soundness. Let P^* output an accepting proof π_i for (z_0, z_i). We construct an extractor that recovers a sequence of indices and witnesses explaining the accepted chain. At the last accepted step, simulation extractability of $\mathsf{NIZK}_{\mathsf{plonk}}$ extracts a witness for the Plonkish relation, including idx, w_{i-1}, the active aggregation proof, and the passive aggregation proofs. The extracted relation enforces $z_i = F_{\mathsf{idx}}(z_{i-1}, \mathsf{w}_{i-1})$, the public hash binding of x_{i-1} and x_i, and successful aggregation verification for all ℓ slots. Extractability of the underlying PCS aggregation then gives openings for the accumulated committed Plonkish instances, except with the PCS soundness error. Applying the same argument inductively to the predecessor proof yields witnesses $(\mathsf{w}_0, \ldots, \mathsf{w}_{i-1})$ and indices $(\mathsf{idx}_0, \ldots, \mathsf{idx}_{i-1})$ such that $z_{t+1} = F_{\mathsf{idx}_t}(z_t, \mathsf{w}_t)$ for every $t \in \{0, \ldots, i-1\}$. Thus any accepting proof that does not admit such a trace breaks either simulation extractability of $\mathsf{NIZK}_{\mathsf{plonk}}$, soundness of the PCS aggregation, or knowledge soundness of the committed Plonkish relation.

Zero knowledge. Zero knowledge is proved by a standard hybrid over the polynomially many steps and slots. First replace each Plonkish proof by the simulator for $\mathsf{NIZK}_{\mathsf{plonk}}$ on the same public statement. Then replace each hidden KZG commitment in the accumulated slots by a hiding commitment to a dummy polynomial of the same degree bound. The public hash bindings are recomputed on the simulated public commitments, and the simulated Plonkish proof enforces the resulting public statement without using the private witness. Each hybrid step is indistinguishable by either zero knowledge of $\mathsf{NIZK}_{\mathsf{plonk}}$ or hiding of the PCS.

Function hiding. We construct a simulator Sim that produces π_{i+1} without using the real function index idx. On input $(\mathsf{vk}, (z_0, z_i), i, z_{i+1})$, Sim samples fresh public accumulator entries for all ℓ slots with the same distribution as dummy aggregated entries. For every slot, the simulated public entry is a fresh hiding commitment to a valid dummy committed Plonkish pair, using the public commitment parameters in the CRS. The simulator then invokes the zero-knowledge simulator for $\mathsf{NIZK}_{\mathsf{plonk}}$ on the public statement $(\mathsf{vk}_{\mathsf{KZG}}, \vec{\mathsf{X}}_i, \mathsf{x}_i, \vec{\mathsf{X}}_{i+1}, (i, z_0, z_i, z_{i+1}))$. It outputs $\pi_{i+1} := ((\vec{\mathsf{X}}_{i+1}, \vec{\mathsf{W}}_{i+1}), (\mathsf{x}_{i+1}, \mathsf{w}_{i+1}))$, where the witness part is simulated consistently with the public dummy slots.

We compare the real and simulated distributions. In the real proof, all ℓ slots are updated at every step: the active slot aggregates the real transition, and every passive slot aggregates the fixed dummy pair $(\mathsf{x}_\perp, \mathsf{w}_\perp)$. In the simulated proof, all ℓ slots are also fresh hiding commitments to dummy committed instances. By the hiding property of KZG, the public committed values in the real active slot are indistinguishable from the simulated dummy committed values. The same holds trivially for the passive slots, which are dummy in both worlds. Finally, the function index appears only inside the Plonkish witness, and replacing the Plonkish proof by its simulator is indistinguishable by zero knowledge of $\mathsf{NIZK}_{\mathsf{plonk}}$. A hybrid over the ℓ slots and the Plonkish proof therefore shows that the adversary cannot distinguish the real active index from the simulated one.

The argument crucially depends on updating every public slot at every step. If a passive public slot were left unchanged, an adversary could compare consecutive accumulator vectors and identify the unique changed slot directly. The construction and the implementation therefore preserve function hiding only when passive slots receive fresh dummy public updates. □

Appendix C. Instantiations of PoP for BLS Signatures

Following Construction 2, we employ the proof composition technique (the "Sigma-plus-IVC" paradigm [13], [14]) to handle expensive algebraic operations: costly operations such as pairings are performed outside the NIZK circuit and proved separately via a Σ-protocol. Since the IVC scheme is treated as a black box, we focus on the Σ-sub-protocol. The main technical challenge lies in hiding both the original signature $\sigma \in \mathbb{G}_1$ and the auxiliary element $R \in \mathbb{G}_1$, which requires committing to group elements rather than field elements. To overcome this, we follow Woo *et al.* [14] and employ ElGamal commitments to hide σ and R.

Definition 12 ([14])**.** *An ElGamal commitment for a group element $G \in \mathbb{G}_1$ is a pair $(C^{(0)}, C^{(1)}) \in \mathbb{G}_1 \times \mathbb{G}_1$ as*

$$(C^{(0)}, C^{(1)}) \coloneqq (\mathsf{Hash}(v, o),\ H^v \cdot G),$$

where $v, o \leftarrow\!\!\$\ \mathbb{F}$ and $H \in \mathbb{G}_1$ is a fixed generator with unknown discrete logarithm relative to the standard generator of $\mathbb{G}_1$. Assuming DDH is hard over $\mathbb{G}_1$, the ElGamal commitment is perfectly hiding and computationally binding.

C.1. Proof of Possession

We present the Σ-sub-protocol for proving possession of a valid BLS signature, *i.e.*, the relation

$$\mathcal{R}_\Sigma = \{(vk, C_\sigma, C_c;\ \sigma, c) \colon e(\sigma, g_2) = e(c, vk)\}\,,$$

combined with Sigmabus's transform [13] to bind the witnesses to the IVC part. The approach modifies the Σ-protocol so that the prover's first message includes a ZKP-friendly commitment to the witnesses, and then the prover additionally proves: (1) these witnesses are the same as those given to the IVC prover, and (2) the Σ-protocol response was computed correctly from these witnesses.

Construction 3 (Σ-Protocol for PoP of BLS Signatures)**.** *Denote $C_\sigma \coloneqq H^{v_0} \cdot \sigma$ and $C_c \coloneqq H^{v_1} \cdot c$, the prover and verifier interact as follows.*

- *The prover takes inputs* pp *including $H \in \mathbb{G}_1$ and samples $s_0, s_1, v_0, o_0, v_1, o_1 \leftarrow\!\!\$\ \mathbb{F}$ and $s_2, s_3 \leftarrow\!\!\$\ \mathbb{G}_1$.*
- *The prover computes extra commitments for cross terms:*

$$\begin{aligned} U_e &\coloneqq e^{-1}(H^{v_1}, vk) \cdot e(H^{v_0}, g_2), \\ U_0 &\coloneqq H^{s_0} \cdot s_2, \quad U_1 \coloneqq H^{s_1} \cdot s_3. \end{aligned}$$

- *The prover computes hash commitments to randomness:*

$$H_\sigma \coloneqq \mathsf{Hash}(v_0, s_0, s_2, o_0), \quad H_c \coloneqq \mathsf{Hash}(v_1, s_1, s_3, o_1).$$

- *The prover sends* $(H_\sigma, H_c, U_e, U_0, U_1)$.
- *The verifier samples a random challenge* $x \leftarrow\$ \mathbb{F}$ *and sends it to the prover.*
- *The prover computes the responses:*

$$\begin{aligned} z_0 &\coloneqq s_0 + x \cdot v_0, \quad z_1 \coloneqq s_1 + x \cdot v_1, \\ z_2 &\coloneqq s_2 \cdot \sigma^x, \quad z_3 \coloneqq s_3 \cdot c^x. \end{aligned}$$

- *The prover sends* $z_0, z_1 \in \mathbb{F}$, $z_2, z_3 \in \mathbb{G}_1$ *to the verifier.*
- *The verifier checks:*

$$\begin{aligned} &e(C_\sigma, g_2) \stackrel{?}{=} e(C_c, vk) \cdot U_e, \\ &H^{z_0} \cdot z_2 \stackrel{?}{=} C_\sigma \cdot U_0, \quad H^{z_1} \cdot z_3 \stackrel{?}{=} C_c \cdot U_1. \end{aligned}$$

For the IVC part, the prover calls the interfaces of SUPERHALO. We present the detailed relation $\mathcal{R}_\Pi$:

$$\left\{ \begin{pmatrix} c,\ H_\sigma,\ H_\sigma,\ H_c \\ v_0,\ v_1,\ o_0,\ o_1 \\ s_0,\ s_1,\ s_2,\ s_3 \end{pmatrix} : \begin{array}{l} H_\sigma = \mathsf{Hash}(v_0, s_0, s_2, o_0) \wedge \\ H_c = \mathsf{Hash}(v_1, s_1, s_3, o_1) \wedge \\ c = \mathsf{Hash}(m) \ \wedge C_c = H^{v_1} \cdot c \end{array} \right\}$$

Proof of Lemma eflem:rand-PoP. Completeness follows from the verification equations of the Σ-sub-protocol and completeness of the IVC link proof. For an honestly generated witness, the pairing equation $e(\sigma, g_2) = e(c, vk)$ holds, the commitments satisfy $C_\sigma = H^{v_0} \cdot \sigma$ and $C_c = H^{v_1} \cdot c$, and the responses satisfy the two linear commitment equations checked by the verifier. The IVC relation also verifies because it checks the same committed digest c, the hash relation $c = \mathsf{Hash}(m)$, and the hash commitments H_σ and H_c used by the Σ transcript.

For adaptive knowledge soundness, consider any accepting non-interactive transcript in the random oracle model. By the forking lemma applied to the Fiat–Shamir challenge, two accepting transcripts with the same first message and distinct challenges extract openings of C_σ and C_c, namely group elements σ and c and commitment randomness v_0, v_1. The first verification equation then gives $e(\sigma, g_2) = e(c, vk)$ after canceling the commitment blinding terms. Knowledge soundness of the IVC link proof extracts the value m and proves that the same c satisfies $c = \mathsf{Hash}(m)$ and is the value committed in C_c. Binding of the ElGamal commitment prevents the Σ side and the IVC side from opening C_c to different values, except with negligible probability. Thus an accepting proof yields a witness (σ, c, m) for possession of a valid BLS signature on m.

Zero knowledge follows by simulating the two components jointly. The simulator first replaces C_σ and C_c with hiding commitments to dummy group elements of the correct type. It then programs the Fiat–Shamir challenge and samples responses z_0, z_1, z_2, z_3 so that the three Σ verification equations hold. Finally, it replaces the IVC link proof by the zero-knowledge simulator on the resulting public statement. The simulated transcript is indistinguishable from a real one by hiding of the ElGamal commitments, zero knowledge of the Σ protocol after Fiat–Shamir, and zero knowledge of the IVC proof. □

C.2. Rerandomizability

We describe how to transform the original proof of possession $(C_\sigma, C_c, \pi_\Sigma, \pi_\Pi)$ into a new proof $(C'_\sigma, C'_c, \pi'_\Sigma, \pi'_\Pi)$ for the same signature (vk, σ, c). The Rerand algorithm computes $C'_\sigma = C_\sigma \cdot H^{v'_0}, C'_c = C_c \cdot H^{v'_1}$ and recomputes π'_Σ for $\mathcal{R}_\Sigma$ with respect to (C'_σ, C'_c).

Appendix D. Full Proofs for Section 5.3

This appendix gives the full proofs for Section 5.3.

Proof of Theorem 1. Fix any PPT adversary $\mathcal{A}$ and consider the experiment $\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda)$ from Fig. 3. Conditioned on the experiment not aborting early, every command in the schedule is *well-formed*: The predicate is supported and satisfied by the honestly issued attribute vector, and the command is called with a defined reuse state. It suffices to show that the verifier accepts at every iteration $t \in [T]$ except with negligible probability. By construction, Issue computes a valid BLS signature σ on $\vec{a}$. For any call to Present, the challenger invokes Π.Prove for the predicates F_{PoP} and F_ϕ, and then for F_{re}. Completeness of Π makes these proofs verify, correctness of Ψ makes the honestly generated commitments C_c, C_σ open to the intended values, and completeness of RPoP accepts the valid signature witness and its commitments linked to the NIVC digest. Thus, the Σ component, the rerandomization consistency check, and all Verify checks are accepted. For Update, the prover extends the NIVC chain with a satisfied predicate ϕ_t and rerandomizes via F_{re}; completeness of Π and correctness of RPoP preserve validity. The same argument applies to Rerand, which performs only rerandomization. A union bound over at most polynomially many steps T preserves negligible error. Hence $\Pr[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(\lambda) = 1] \le \mathsf{negl}(\lambda)$. □

Proof of Theorem 2. Fix a PPT adversary $\mathcal{A}$ attacking $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{A}}(\lambda)$ (Fig. 5) with oracle access (Fig. 4). Let Win be the event that $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{A}}(\lambda) = 1$. Condition on Win. Then $\Omega.\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}_\Pi, \mathsf{vk}, \mathsf{pt}^*, \phi^*_1, \ldots, \phi^*_n, \mathsf{ctx}^*) = 1$ and $\mathsf{pt}^* \notin \mathcal{Q}_{\mathsf{Tok}}$, and no revealed credential in $\mathcal{Q}_{\mathsf{Rev}}$ satisfies all predicates. Here $\mathcal{O}_{\mathsf{Issue}}$ returns only opaque handles; credentials are exposed only through $\mathcal{O}_{\mathsf{Reveal}}$, and the holder oracles return honestly generated handle-based tokens. Thus revealed credentials are ruled out by the winning condition, while issued-but-unrevealed credentials are covered by q-RT-CDH.

(1) *Extraction from the accepting token.* Since Verify accepts, the NIVC and RPoP sub-checks accept. Knowledge soundness of Π gives, except with probability bounded by $\mathsf{Adv}^{\mathsf{ks}}_\Pi$, witnesses for the predicate and rerandomization chain, including an attribute vector $\vec{a}^*$ such that $\forall j \in \{1, \ldots, n\}\colon \phi^*_j(\vec{a}^*) = 1$. Adaptive knowledge soundness of RPoP gives, except with probability bounded by $\mathsf{Adv}^{\mathsf{ks}}_{\mathsf{RPoP}}$, a witness $(\widetilde{\sigma}^*, \widetilde{c}^*, \rho^*)$ with $\rho^* \neq 0$ and $e(\widetilde{\sigma}^*, g_2) = e(\widetilde{c}^*, \mathsf{vk})$. The F_{PoP} step and RPoP link proof bind the ElGamal opening in the Σ component to the digest carried by the

NIVC chain; hence, except with negligible probability, $\widetilde{c}^* = \mathsf{Hash}(\vec{a}^*)^{\rho^*}$. Let Bad_Π (resp. $\mathsf{Bad}_{\mathsf{RPoP}}$) be the event that extraction from Π (resp. from RPoP) fails on an accepting token. Then $\Pr[\mathsf{Win}] \le \Pr[\mathsf{Win} \wedge \neg\mathsf{Bad}_\Pi \wedge \neg\mathsf{Bad}_{\mathsf{RPoP}}] + \Pr[\mathsf{Bad}_\Pi] + \Pr[\mathsf{Bad}_{\mathsf{RPoP}}]$. The last two terms are bounded by the corresponding knowledge-soundness advantages.

(2) *Reduction to signature security.* Consider the event $\mathsf{Win} \wedge \neg\mathsf{Bad}_\Pi \wedge \neg\mathsf{Bad}_{\mathsf{RPoP}}$. By the extracted relations above, $e(\widetilde{\sigma}^*, g_2) = e(\mathsf{Hash}(\vec{a}^*)^{\rho^*}, \mathsf{vk})$. Since $\rho^* \neq 0$, let $\sigma^* = (\widetilde{\sigma}^*)^{1/\rho^*}$. Then $e(\sigma^*, g_2) = e(\mathsf{Hash}(\vec{a}^*), \mathsf{vk})$, which is the BLS verification equation for message $\vec{a}^*$. By the winning condition, $\vec{a}^*$ cannot equal the attribute vector of any revealed credential. If $\vec{a}^*$ was never signed by the issuer, then $(\vec{a}^*, \sigma^*)$ is an EUF-CMA forgery for Sig. Otherwise, $\vec{a}^*$ belongs to an issued-but-unrevealed credential, and computing $\sigma^* = \mathsf{Hash}(\vec{a}^*)^{\mathsf{sk}}$ after seeing only rerandomized pairs $(\mathsf{Hash}(\vec{a}^*)^r, (\sigma^*)^r)$ violates q-RT-CDH. The reduction simulates oracle proofs with the ZK simulators for Π and RPoP, programs Fiat–Shamir consistently, and uses the q-RT-CDH auxiliary pairs as honest holder outputs. By zero-knowledge, the simulated transcripts are indistinguishable from real ones. Putting the bounds together yields the claimed inequality. □

Proof of Lemma 2. For $r \in \mathbb{F}_p^*$, the map $f_r\colon \mathbb{G} \to \mathbb{G}$ given by $f_r(x) = x^r$ is a permutation. Thus, for uniform $r \leftarrow\$ \mathbb{F}_p^*$, x^r is uniform over $\mathbb{G}$. Sampling $r \leftarrow\$ \mathbb{F}_p$ differs only by the event $r = 0$, which occurs with probability $1/p \in \mathsf{negl}(\lambda)$. □

Proof of Theorems 3 and 4. Fix a PPT adversary $\mathcal{A}$ and consider either $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda)$ (Fig. 6) or $\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A}}(\lambda)$ (Fig. 7). Condition on an admissible challenge. The hidden bit selects the challenge credential in anonymity, or whether two tokens come from the same credential in unlinkability, covering both fresh and flow modes. We use hybrids over the challenge tokens generated by Run.

Game G_0*:* This is the real experiment.

Game G_1 *(simulate* Π *witnesses):* In G_1, every challenge-transcript call to $\Pi.\mathsf{Prove}$ is replaced by the zero-knowledge simulator for Π, on the same public statements and visible schedule as in G_0. By zero-knowledge of Π, $|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \le \mathsf{Adv}^{\mathsf{zk}}_\Pi(\mathcal{B}^{\mathsf{zk}}_\Pi)$. The reduction $\mathcal{B}^{\mathsf{zk}}_\Pi$ embeds its zero-knowledge challenge in these NIVC proofs.

Game G_2 *(hide the* Π *function trace):* In G_2, we keep the simulated NIVC proofs but replace the visible update-function trace with the ideal challenge trace, preserving the public step count. By function-hiding of Π, $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \le \mathsf{Adv}^{\mathsf{fh}}_\Pi(\mathcal{B}^{\mathsf{fh}}_\Pi)$. The reduction $\mathcal{B}^{\mathsf{fh}}_\Pi$ embeds its challenge in the visible NIVC transcript. This hybrid is valid for fixed-length or explicitly padded challenge transcripts; if two challenge worlds expose different numbers of NIVC steps, the gap is charged to an iteration-hiding assumption (which SUPERHALO satisfies by the argument [19]).

Game G_3 *(simulate RPoP proofs for real statements):* In G_3, every composed RPoP proof in the challenge transcript is replaced by the RPoP zero-knowledge simulator on the *real* public statements $(vk, C_\sigma, C_c, \ldots)$. After this step, composed PoP proofs use no witness (σ, c). By zero-knowledge of RPoP, $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]| \le \mathsf{Adv}^{\mathsf{zk}}_{\mathsf{RPoP}}(\mathcal{B}_{\mathsf{RPoP}})$.

Game G_4 *(replace commitments and re-simulate RPoP proofs):* In G_4, each commitment under Ψ to credential-dependent values (C_σ, C_c) is replaced by a commitment to a fixed dummy value of the same type, yielding (C^*_σ, C^*_c). We also re-run the RPoP ZK simulator on the new public statements $(vk, C^*_\sigma, C^*_c, \ldots)$. This is valid because, after G_3, the simulated RPoP proof depends only on its public statement; any distinguisher for the replacement breaks hiding of Ψ. By a standard sequence of hybrids over all commitments in the experiment, $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_4 = 1]| \le \mathsf{Adv}^{\mathsf{hide}}_\Psi(\mathcal{B}_\Psi) + \mathsf{negl}(\lambda)$.

Game G_5 *(replace rerandomized digest values):* In G_5, each rerandomized digest value $z_i = c^r$ in a challenge token is replaced by an independent uniform element of $\mathbb{G}_1$. The simulated NIVC and RPoP proofs can be re-run on these public statements. By Lemma 2, the original values are statistically close to uniform and independent of the credential. Therefore, $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_5 = 1]| \le \mathsf{negl}(\lambda)$.

Concluding the hybrid argument. In G_5, challenge tokens contain only simulated NIVC proofs, re-simulated RPoP proofs, dummy commitments, and uniform group elements. The transcript therefore depends only on public schedules, predicates, and contexts, not on the hidden bit. Hence $\Pr[\mathsf{G}_5 = 1] = 1/2$. Combining the hybrid bounds proves both theorems. □