Polynomial Batch Accumulators for Onboarding-Dominant Zero-Knowledge Membership Proofs

Samanta Karmaker
Brain Station 23 PLC
Dhaka, Bangladesh
samonto0726@gmail.com

Sampad Sikder
Brain Station 23 PLC
Dhaka, Bangladesh
bsse1219@iit.du.ac.bd

Md Ariful Islam
Brain Station 23 PLC
Dhaka, Bangladesh
fahim.arif0373@outlook.com

Md Sadek Ferdous
BRAC University
Dhaka, Bangladesh
sadek.ferdous@bracu.ac.bd

***Abstract*—Zero-knowledge membership accumulators are conventionally tailored for slowly growing sets with frequent proof generation, creating severe efficiency bottlenecks for modern compliance architectures. Governance-driven applications like FATF customer due diligence, eIDAS 2.0 wallet issuance, and MiCA VASP authorization invert this structure. In this regime, rosters are ingested at an enterprise scale, while proofs are requested selectively. To resolve this mismatch, we introduce Polynomial Equation Batching (PEB), a membership accumulator for onboarding heavy workloads. PEB represents up to 128 members per batch as a characteristic polynomial and hashes its coefficients via Poseidon, seamlessly scaling to arbitrarily large populations through parallel batching. We present two variants: PEB-A, which maps member secrets directly to polynomial roots for maximum performance, and PEB-B, which hashes secrets in-circuit for hardened security. Implemented in Noir and Barretenberg on BN254, and evaluated for up to 1,048,576 members, PEB provides significant performance improvements. Compared to incremental Merkle trees, PEB-A achieves up to a $149.07\times$ mean speedup in per-user initialization while incurring a modest 33.87% proof-generation overhead. Additionally, against static Merkle trees, PEB-A delivers a $15.71\times$ speedup during setup. By shifting off-circuit overhead from hash evaluations to field multiplications, PEB provides a highly efficient, scalable solution for real-world zero-knowledge compliance systems.**

***Keywords*—**privacy-preserving protocols, pseudonymity and anonymity, zero-knowledge proofs, cryptographic accumulators, set membership, zk-SNARKs, polynomial batching, Merkle trees, Poseidon, regulatory compliance

I. Introduction

Cryptographic accumulators are a foundational primitive for zero-knowledge membership proofs, allowing a user to prove membership in a committed set without revealing their identity. This capability is anchored by privacy pools [1], anonymous credential systems [2], and a growing class of compliance-sensitive financial protocols deployed on public blockchains. As regulated financial services transition to public networks such as Ethereum, institutions migrating from permissioned infrastructure increasingly seek public deployments to leverage shared liquidity and interoperability [3]. Consequently, the need for accumulators that can scale to millions of approved members has become a practical production challenge.

The current dominant accumulator construction is the Merkle tree, which hashes members into a binary tree and exposes only the root as public state. A Merkle-based accumulator can be populated incrementally, member by member (Incremental Merkle Tree, IMT), or in bulk from a complete roster (Static Merkle Tree, SMT)[1] built in a single pass. An IMT recomputes a full leaf-to-root path on each insertion, which costs $\mathcal{O}(\log n)$ hashes per member for a tree accumulating n members. An SMT hashes each internal node once, for $\mathcal{O}(n)$ hashes in total, though it still requires the full roster in advance and concentrates cost upfront. Both constructions are optimized for the case where proof generation, not member ingestion, dominates system cost.

However, many production-grade compliance systems invert this cost profile. Governance-defined rosters of authorized member public keys or identity commitments are ingested in bulk and updated on discrete epochs across several regulatory regimes. These include customer due diligence under Financial Action Task Force (FATF) Recommendation 10 [4], electronic IDentification, Authentication and trust Services (eIDAS) 2.0 onboarding [5], and Markets in Crypto-Assets (MiCA)-aligned Virtual Asset Service Provider (VASP) authorization [6], [7]. Proofs against these rosters are then generated selectively and infrequently. Here, onboarding throughput is the binding bottleneck, and even SMT's linear hashing cost can be expensive at scale. Existing work on bilinear accumulation batching [8] and RSA-based proofs [9] addresses adjacent problems but does not target this onboarding-dominant regime.

We introduce Polynomial Equation Batching (PEB), a framework designed specifically for this operational regime. We encode a bounded batch of up to $b = 128$ members as the roots of a characteristic polynomial over a finite field. Membership is evaluated in-circuit via Horner's method [10] in a fixed 129 multiply-add steps, independent of total system scale. Once a batch fills, subsequent members instantiate a new batch using the identical compiled circuit and proving keys, publishing each commitment as an independent on-chain event.

[1]Not to be confused with the Sparse Merkle Tree, also abbreviated SMT in parts of the literature. We use SMT exclusively for the bulk-construction strategy.

The key differentiator is the cost of user ingestion. A Poseidon evaluation costs many field multiplications, while extending a batch polynomial by one member costs one pass of field multiply-accumulate operations over the coefficient vector. On the contrary, Merkle ingestion require at least 21 hash evaluations per member at depth 20. The speedup reflects the native off-circuit cost of a Poseidon evaluation versus a field multiplication in our implementation.

However, relying directly on unhashed polynomial roots introduces algebraic security risks, like full-vector coefficient leakage via root-finding and state-change exploitation via Greatest Common Divisor (GCD) algorithms. Therefore, we present two variants. PEB-A maximizes speedups under a host-level confidentiality assumption on the coefficient vector. PEB-B hashes member secrets in-circuit, making high-entropy secrets immune to such attacks with a relatively modest speedup. Both variants operate in an issuer-as-prover architecture, in which the issuer retains the batch coefficient vector and constructs membership proofs (§III).

We implemented both variants using Noir [11] with the Barretenberg [12] backend on BN254 and evaluated them against IMT and SMT baselines across 14 population sizes ranging from 2^7 (128) to 2^{20} (1,048,576) members. In our primary environment, PEB-A achieved a mean per-user initialization speedup of $149.07\times$ over IMT, at a 33.87% mean proof-generation overhead. Against SMT, its speedup was $15.71\times$, at a 30.44% overhead. The hardened variant, PEB-B, achieved $18.18\times$ over IMT (30.11% overhead) and $1.80\times$ over SMT (31.68% overhead). Our results show that PEB is preferable when onboarding throughput is the binding constraint and an issuer-as-prover model is acceptable. Conversely, Merkle trees remain better suited when proof-generation frequency dominates total system cost or when strict self-sovereign witness custody is required.

A. Contributions

1) **The PEB Construction:** We introduce a polynomial-based membership accumulator for onboarding-dominant systems. We provide formal specifications for two variants (PEB-A and PEB-B) that balance performance against algebraic confidentiality assumptions.
2) **Security Analysis:** We characterize the algebraic attack vectors against polynomial-encoded sets, namely full-vector coefficient leakage and multi-state polynomial division. We state the corresponding mitigations for both variants. We prove correctness, soundness, and zero-knowledge for both variants in a threat model with an honest issuer. We also prove state binding, append-only consistency, public verifiability, and revocation feasibility in principle.
3) **Empirical Evaluation:** We implement PEB in Noir [11]/Barretenberg [12] and evaluate it against IMT and SMT baselines up to 1,048,576 members in two environments. We quantify trade-offs among initialization throughput, proof latency, proof size, storage, on-chain footprint, and witness custody.
4) **Deployment Guidance:** We qualitatively discuss how PEB's architectural trade-offs relate to representative regulatory identity and compliance contexts, including eIDAS 2.0, MiCA, and FATF-aligned due diligence.

II. Background and Technical Overview

A. Terminology

- **Epoch:** A snapshot period with a fixed membership set.
- **Commitment:** A public value, published on-chain, that binds a batch.
- **Witness:** The private input to a proof. In the issuer-as-prover model, the issuer holds it without distributing it.
- **Batch closure:** Publishing a commitment for a batch's current members. A new commitment supersedes the previous one, and proofs against the old one are rejected.
- **Transparency:** No trusted setup or algebraic trapdoor in the accumulator layer. The SNARK layer uses a universal Structured Reference String (SRS) shared by all constructions benchmarked.

B. Zero-Knowledge Proof Systems

A zero-knowledge proof is a protocol between a prover and verifier for a language L. It satisfies completeness (an honest prover convinces an honest verifier for any $x \in L$), soundness (no witness for $x \notin L$ is accepted except with negligible probability), and zero-knowledge (the verifier's view is simulatable without the witness) [13]. We assume standard zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [14], [15] over arithmetic circuits. Constraint count governs prover time, and verifier time is constant. We instantiate our construction in Noir [11], compiled to the Abstract Circuit Intermediate Representation (ACIR) and proven via Barretenberg [12] using UltraHonk, a PLONK-family system [15], over BN254. All field elements lie in the scalar field $\mathbb{F}_p$ of BN254. We further rely on simulation-extractability. This property guarantees that, for an adversary with adaptive access to a simulation oracle, an extractor fails to recover a valid witness from a new adversarial proof with probability at most $\varepsilon_{\mathsf{SE}}$. We assume that UltraHonk satisfies this property, following the analysis of Plonk-family constructions [16].

C. Cryptographic Accumulators

A cryptographic accumulator is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Add}, \mathsf{GenProof}, \mathsf{Verify})$ that commits a set S to a concise public value C. For any $s \in S$, a short witness π can be generated and verified against C in time independent of $|S|$ [17], [18]. Our construction refines this syntax with an explicit Commit algorithm (§IV). Hash-based tree accumulators require only collision resistance but have logarithmic proof sizes. RSA-based [19] and pairing-based [20] accumulators achieve constant-size proofs, but rely on non-standard algebraic assumptions or a trusted setup.

D. Merkle Trees

A Merkle tree over a set S is a binary tree in which each leaf holds $\mathsf{Hash}(s)$ for $s \in S$ and each internal node holds the hash of its two children. The root rt commits to S, and membership of a leaf is provable through the sibling hashes on its root path [21]. For a tree sized to n members, the depth is $\mathcal{O}(\log n)$. Merkle trees underlie Semaphore [22], Tornado Cash [23], and 0xbow Privacy Pools [24]. We evaluate two population strategies over this structure: an Incremental Merkle Tree (IMT), which inserts members sequentially, and a bulk bottom-up tree (SMT), built in one pass. For a tree sized to n, IMT incurs $\mathcal{O}(n \log n)$ hashes and SMT incurs $\mathcal{O}(n)$. Even arithmetic-optimized hashes like Poseidon require many field multiplications per evaluation [25], making large-scale roster ingestion expensive.

E. Our Approach: A Polynomial-Based Membership Encoding

PEB replaces the binary tree with a polynomial encoding. We encode a batch of up to $b = 128$ members as the roots of a monic polynomial over $\mathbb{F}_p$,

$$P(x) = \prod_{i=1}^{m}(x - s_i) = c_0 + c_1 x + \cdots + c_{m-1}x^{m-1} + x^m$$

$$m \leq b$$

where $s_1, \ldots, s_m$ are member secrets. The coefficient vector is $\mathbf{c} = (c_0, \ldots, c_b)$ with $c_m = 1$ and $c_j = 0$ for $j > m$. The two constructions trade onboarding speed for hardened security by hashing member secrets with Poseidon2 before they become roots (§IV-D–§IV-E). Expanding the polynomial through sequential multiplication takes $\mathcal{O}(b^2)$ field operations per batch. Since b is a compile-time constant, this gives an amortized cost of $\mathcal{O}(1)$ with respect to the total system size n.

Population speed comes from this algebraic representation. Extending a batch with a new member via `Add` takes $\mathcal{O}(b)$ field multiply-accumulate operations. The speedup reflects the native, off-circuit cost of a Poseidon evaluation versus a field multiplication. Membership verification checks $P(s) = 0$ in-circuit using Horner's method [10], which takes $b + 1 = 129$ multiply-add steps. We defer the full protocol, including the issuer-as-prover model and the on-chain binding, to §IV-F.

The batch size m need not equal b. When $m < b$, unused higher-degree coefficients are padded with zeros to keep the vector length at $b + 1$. The public accumulator state is a single Poseidon hash over $(c_0, \ldots, c_b)$. The issuer computes it at each commit, and a later commit supersedes an earlier one. A batch remains appendable up to capacity b. Adding a member a updates $P'(x) = P(x)(x - a)$ without retaining historical coefficients. The issuer then publishes the new commitment and regenerates proofs from the new vector on request.

Populations exceeding one batch are partitioned by $\mathsf{Assign}(ref) = \lfloor \mathrm{index}(ref)/b \rfloor$. This gives $\mathcal{O}(1)$ batch lookups and an $\mathcal{O}(n/b)$ on-chain commitment footprint. The key differences from Merkle-based methods are initialization throughput and witness custody, formalized in §III.

III. Problem Statement and Design Goals

A. Architectural Design Axes

The design space for zero-knowledge membership systems is governed by two independent choices: which system lifecycle phase dominates computational cost, and how witness state is maintained.

a) Axis 1 (Cost Dominance): Systems operate in either a *proof-dominant* or an *onboarding-dominant* regime. In a proof-dominant regime, proof generation occurs frequently while user ingress is rare. Conversely, in an *onboarding-dominant regime*, such as compliance-driven enterprise deployments governed by FATF Recommendation 10 [4], Financial Crimes Enforcement Network (FinCEN) due diligence, and eIDAS 2.0, the primary bottleneck is bulk roster ingestion and vetting, with proofs generated selectively against established rosters.

b) Axis 2 (Witness Custody): The commitment structure dictates witness distribution. It requires either individual, self-custodial membership witnesses maintained per user or a consolidated witness state held by a designated proving entity.

PEB is explicitly engineered for the *high-throughput onboarding, custodial-witness* point in this space.

B. Structural Bottlenecks of Merkle Trees in Onboarding-Dominant Regimes

Merkle tree-based solutions are poorly suited for onboarding-dominant systems due to their heavy hash evaluation cost:

- **Streaming IMTs** pay a *cumulative* tax. Each insertion recomputes a full leaf-to-root path. At our fixed depth of 20, this is 21 hashes per insertion, so ingesting n members takes $21n$ sequential hash evaluations. For an IMT designed to accumulate n members, this is $\mathcal{O}(n \log n)$.
- **Bulk SMTs** front-load this cost. A single bottom-up build pass requires $\mathcal{O}(n)$ hashes, which concentrates severe computational latency onto the epoch boundary.

This initialization latency translates directly into compliance enforcement gaps. Regulatory compliance systems are inherently snapshot-driven. For example, the U.S. Department of the Treasury's Office of Foreign Assets Control (OFAC) Specially Designated Nationals (SDN) list [26] and United Nations (UN) Security Council sanctions lists [27] are released as versioned, discrete manifests. Virtual Asset Service Providers (VASPs) correspondingly reconcile roster state in periodic bulk updates [6]. As demonstrated by Sarkhosh et al. [28], even brief update lags are actively exploited via chain-hopping, fragmented fund flows, and cross-chain bridging. Privacy pools [1] suffer from the same exposure window between bad-actor identification and formal exclusion.

In these regimes, the primary architectural bottleneck is therefore not proof-generation speed, but initialization and update latency. This motivates an algebraic construction like PEB that batch-encodes membership at a fraction of the per-member ingestion cost of Merkle trees.

C. Design Goals and Decisions

We target environments with three simultaneous properties: **(i) Batch-Bounded** (each batch closes with a fixed maximum membership, regardless of streaming or known-in-advance arrival), **(ii) Onboarding-Dominant** (Axis 1: cost is driven by ingestion throughput, not proof frequency), and **(iii) Issuer-as-Prover** (Axis 2: a trusted intermediary proves membership on a member's behalf). The three design goals below follow directly from these properties in order of constraint.

P1: Batch-at-commit optimization. Elements accumulate locally within a bounded batch. `Add` updates the polynomial via $\mathcal{O}(b)$ field arithmetic by computing $P'(X) = P(X)(X-a)$ on the coefficient vector, deferring hashing to a single Poseidon evaluation per `Commit`. This resolves both Merkle failure modes from §III-B. It replaces the path recomputations of IMT with local operations, and it replaces the front-loaded tree build of SMT with one vector hash. Decoupling accumulation from publication is unviable in streaming systems, as continuous mutation frequently invalidates previous witnesses.

Because `Add` updates $\mathbf{c}$ purely off-chain without modifying public state, state transitions occur only when the operator invokes `Commit`, which emits a `BatchCommitted` event with $h = Poseidon(\mathbf{c})$ (§IV-F). On-chain verifiers evaluate proofs strictly against published commitments h. Existing members therefore verify against the active h, while new off-chain members become verifiable only after the next `Commit`.

> **Consequence:** `Add` accumulates coefficients off-chain, deferring public state updates to `Commit`. The issuer retains $\mathbf{c}$ as a custodial prover witness without distributing it to members.

P2: Sufficiency of epoch-level freshness. The custodial model required by Axis 2 is viable because onboarding is the primary focus. The accumulator only needs to ensure "valid membership at epoch v," rather than providing an immediate confirmation of real-time status. This approach is operationally and legally adequate, as regulatory status changes happen in discrete cycles.

> **Consequence:** Epoch transitions happen only when governance creates a new roster. A single batch rebuild takes milliseconds (§V).

P3: Alignment with existing operator trust. The custody requirements for Axis 2 and P2 do not impose new burdens. Trusted intermediaries, such as compliance departments, regulators, and governance bodies, already generate compliance rosters. We therefore adopt an issuer-as-prover model. It adds no trust beyond what the issuer already holds for roster compilation. It does add an availability dependence and issuer-side visibility of every verification request (§IV-F). This approach satisfies Axis 2's custody requirement. It also prevents algebraic attacks (§IV-H1–IV-H2) because the coefficient vector is never exposed publicly. Furthermore, this custody boundary enables two trade-off variants. One leverages the boundary for faster speed (PEB-A). The other removes reliance on it for weaker trust models (PEB-B).

> **Consequence:** The optimized PEB-A (§IV-D) operates under a *Private Polynomial Assumption* (Assumption 1). This is appropriate for regulated Virtual Asset Service Providers (VASPs), permissioned Decentralized Autonomous Organizations (DAOs), or Trusted Execution Environment (TEE)-backed services. The hardened PEB-B (§IV-E) eliminates this assumption for environments without such guarantees.

Non-Goals. We explicitly exclude the following from this work's scope:

- *Self-sovereign proving:* in the default model, members cannot generate proofs without a custodial intermediary. Appendix A-E analyzes this non-default variant.
- *Per-insertion public commitment:* P1 defers publication to `Commit`.
- *Variable batch capacity:* $b = 128$ is a compile-time constant. Adaptive sizing is future work.
- *Benchmarked deletion:* Appendix A shows that removal via epoch regeneration is feasible in principle. Direct removal via polynomial division is simpler but unformalized and unbenchmarked (analyzed in §IV-H2). Revocation propagation to verifiers is out of scope.
- *Distributed maintenance:* a single operator maintains each batch. Witness distribution is out of scope.

D. Formal Threat Model

Roles and message flow follow Figure 2 (§IV-F), and the algorithms are those of Definition 4.

State. The issuer $\mathcal{I}$ holds a secret s_{ref} for each member reference $ref \in \mathsf{Refs}$, together with the coefficient vector $\mathbf{c}$ of that member's batch. For a commitment h on the public log $\mathcal{L}$, let S_h be the secrets of the batch it commits to. The secrets $S = \{s_{ref}\}$ and all vectors $\mathbf{c}$ are known only to $\mathcal{I}$. For a member reference ref and verifier key k_V, the public instance is $x_{\mathsf{pub}}(ref, k_V) = (h, \nu, k_V)$. Here h is the commitment of the member's batch and $\nu = Poseidon(s_{ref}, k_V)$. The private witness is $x_{\mathsf{priv}}(ref) = (s_{ref}, \mathbf{c}, \mathsf{isKYCed})$.

Trust assumptions. We assume a universal, updatable SRS and an honest issuer that builds commitments correctly, attests eligibility, and serves proofs only to authenticated verifiers. $\mathcal{L}$ authenticates the poster of each commitment. S and $\mathbf{c}$ never leave $\mathcal{I}$'s infrastructure. PEB-A requires this (Assumption 1) and PEB-B keeps it only as defense-in-depth.

Adversary. $\mathcal{A}$ is a probabilistic polynomial-time (PPT) adversary that sees pk, vk, $\mathcal{L}$, and Refs, but never S or any $\mathbf{c}$. It may act as a verifier with its own registered keys $K_{\mathcal{A}}$. It makes at most $q = \mathsf{poly}(\lambda)$ adaptive queries to $O_{\mathsf{GenProof}}(ref, k_V)$ with $ref \in \mathsf{Refs}$ and $k_V \in K_{\mathcal{A}}$. It then outputs $(x^*_{\mathsf{pub}}, \pi^*)$ with $x^*_{\mathsf{pub}} = (h^*, \nu^*, k^*_V)$.

Adversary goals. $\mathcal{A}$ seeks to (i) win $\mathsf{Exp}_{\Pi,\mathcal{A}}$ (soundness, Def. 2), or (ii) recover some s_{ref} from public values and oracle outputs. Goal (ii) lies outside the experiment because $\mathcal{A}$ never

Experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}(1^\lambda)$
1: $(pk, vk) \leftarrow \mathsf{Setup}(1^\lambda, b)$, $Q \leftarrow \emptyset$
2: $(x^*_{\mathsf{pub}}, \pi^*) \leftarrow \mathcal{A}^{O_{\mathsf{GenProof}}}(pk, vk, \mathcal{L}, \mathsf{Refs})$

Oracle $O_{\mathsf{GenProof}}(ref, k_V)$ (at most q calls):
3: **if** $ref \notin \mathsf{Refs} \vee k_V \notin K_{\mathcal{A}}$ **then return** $\perp$
4: $\pi \leftarrow \mathsf{GenProof}\big(pk,\ x_{\mathsf{pub}}(ref, k_V),\ x_{\mathsf{priv}}(ref)\big)$
5: $Q \leftarrow Q \cup \{(x_{\mathsf{pub}}(ref, k_V), \pi)\}$, **return** $(x_{\mathsf{pub}}(ref, k_V), \pi)$

$\mathcal{A}$ **wins iff:** (a) h^* is posted on $\mathcal{L}$, (b) $\mathsf{Verify}(vk, x^*_{\mathsf{pub}}, \pi^*) = 1$, (c) $(x^*_{\mathsf{pub}}, \pi^*) \notin Q$, and (d) the extracted witness $s' \leftarrow \mathcal{E}(x^*_{\mathsf{pub}}, \pi^*)$ has $s' \notin S_{h^*}$.

Fig. 1: Soundness experiment. $\mathcal{E}$ is the SNARK's simulation-extractor. For PEB-B, membership is tested on $Poseidon(s')$.

sees **c**. Leakage of **c** through a custody failure is analyzed in §IV-H.

Cryptographic assumptions. Security relies on (i) collision, and preimage resistance of Poseidon2, (ii) keyed indistinguishability of Poseidon2, and (iii) zero-knowledge and simulation-extractability of the zk-SNARK. Keyed indistinguishability is a non-standard assumption. It means that for an unknown high-entropy s and adversarial verifier keys $k_1, \ldots, k_q$ the outputs $Poseidon(s, k_i)$ look independent and random. We use it only for nullifier unlinkability. We make no claim of membership-set hiding, side-channel resistance, or post-quantum security.

Applicability Framework. Merkle trees remain the preferred choice when self-sovereign proof generation is essential. Within PEB, PEB-A cannot support self-sovereign proving because holding the coefficients directly exposes the secret (§IV-H1–IV-H2). PEB-B can technically allow it (Appendix A-E), but it leaves residual, entropy-dependent exposure, so we do not recommend it as a default. Bridging this witness-custody gap remains open for future work (§VIII-A).

IV. Construction and Security Analysis

A. Common Components

a) Hash Function: We use Poseidon [29] as the zk-friendly hash primitive for commitments, hashed roots, and nullifiers.

b) KYC Flag: A binary flag indicates whether a member meets Know Your Customer (KYC) requirements. The proof logic enforces it. It represents the issuer's assertion that the holder is a KYC-verified member. The predicate is a placeholder at the application layer and can be customized for each deployment.

c) Nullifier: Constructions also support an optional nullifier and verifier-key mechanism (§IV-F).

These components are identical across IMT, SMT, and PEB, giving a consistent baseline for comparison.

B. Security Definitions

We formalize security with respect to the accumulator model of §II-C. We use the following generic syntax. $\mathsf{Setup}(1^\lambda) \to (pk, vk)$ outputs proving and verification keys. $\mathsf{Commit}(S) \to C$ commits a set $S \subset \mathbb{F}_p$. $\mathsf{GenProof}(pk, x_{\mathrm{pub}}, x_{\mathrm{priv}}) \to \pi$ and $\mathsf{Verify}(vk, x_{\mathrm{pub}}, \pi) \to \{0, 1\}$ prove and check membership. The public instance x_{pub} contains C. The private witness x_{priv} contains a secret $s \in S$ and any auxiliary data. Definition 4 instantiates this syntax for PEB. The following three properties are required.

Definition 1 (Correctness)**.** *A membership accumulator satisfies **correctness** if, for any set S, any $s \in S$, and any honestly generated (pk, vk), C, and π, the proof verifies with overwhelming probability:*

$$\Pr\big[\mathsf{Verify}(vk, x_{pub}, \pi) = 1 \mid \pi \leftarrow \mathsf{GenProof}(pk, x_{pub}, x_{priv})\big] \ \geq\ 1 - \mathsf{negl}(\lambda)$$

Definition 2 (Soundness / Membership Unforgeability)**.** *Let Q be the set of (instance, proof) pairs returned by an honest proof oracle O_{GenProof}. A membership accumulator satisfies **soundness** if, for any PPT adversary $\mathcal{A}$ making at most $q = \mathsf{poly}(\lambda)$ oracle queries, the probability that $\mathcal{A}$ outputs an accepted, fresh proof for an element outside S is negligible:*

$$\Pr\left[\begin{array}{l|l} \mathsf{Verify}(vk, x^*_{pub}, \pi^*) = 1 & (pk, vk) \leftarrow \mathsf{Setup}(1^\lambda),\ C \leftarrow \mathsf{Commit}(S), \\ \wedge\ (x^*_{pub}, \pi^*) \notin Q\ \wedge\ s' \notin S & (x^*_{pub}, \pi^*) \leftarrow \mathcal{A}^{O_{\mathsf{GenProof}}}(pk, vk, C), \\ & s' \leftarrow \mathcal{E}(x^*_{pub}, \pi^*) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Here $\mathcal{E}$ is the simulation-extractor of the SNARK. The experiment of Figure 1 instantiates this definition for PEB. *The adversary does not receive S.*

Definition 3 (Zero-Knowledge)**.** *A membership accumulator satisfies (computational) **zero-knowledge** if there exists a PPT simulator* Sim *such that, for every $s \in S$, an honestly generated proof is computationally indistinguishable from a proof produced by* Sim *given only the public instance x_{pub}:*

$$\big\{\pi : \pi \leftarrow \mathsf{GenProof}(pk, x_{pub}, x_{priv})\big\}\ \approx_c\ \big\{\pi^* : \pi^* \leftarrow \mathsf{Sim}(x_{pub})\big\}$$

The proof π reveals nothing about the witness beyond x_{pub} and the truth of the membership statement. The values in x_{pub}, such as the nullifier, are outside this guarantee and are discussed in §IV-F.

Additional properties (state binding, append-only consistency, public verifiability, and revocation feasibility) are established in Appendix A.

C. Merkle Tree Baseline Constructions

We use two Merkle-tree baselines, IMT and SMT, matching the population strategies in §III. They use the same membership-proof circuit but differ in member-insertion strategy.

1) Structure, Commitment, and Population: Members are committed by hashing each secret into a leaf and computing internal nodes bottom-up:

$$N_{0,i} = Poseidon(s_i),$$
$$N_{\ell,j} = Poseidon(N_{\ell-1,\,2j},\ N_{\ell-1,\,2j+1})$$

The root $rt = N_{20,0}$ is the only public commitment, and revealing a leaf provides no information about the underlying member for high-entropy secrets. IMT builds this structure by incremental, per-member insertion. At depth 20, each insertion

Algorithm 1 Merkle Membership Verification Circuit (shared by IMT and SMT)

Require: Public: root rt, nullifier ν, verifier key k_V
Require: Private: secret s, KYC flag `isKYCed`, leaf index i, path $\{p_0, \ldots, p_{19}\}$, directions $\{\delta_0, \ldots, \delta_{19}\}$

assert `isKYCed` $=$ `true` ▷ Step 1: KYC check
$u \leftarrow Poseidon(s)$ ▷ Step 2: Hash secret to leaf
for $j \leftarrow 0$ **to** 19 **do** ▷ Step 3: Walk leaf to root
  if $\delta_j = 1$ **then**
    $(u_L,\ u_R) \leftarrow (p_j,\ u)$
  else
    $(u_L,\ u_R) \leftarrow (u,\ p_j)$
  end if
  $u \leftarrow Poseidon(u_L,\ u_R)$
end for
assert $u = rt$ ▷ Step 4: Root consistency
assert $\sum_{j=0}^{19} \delta_j \cdot 2^j = i$ ▷ Step 5: Index encoding
assert $Poseidon(s,\ k_V) = \nu$ ▷ Step 6: Nullifier check

costs 21 hashes, so the total cost is linear in n (for a tree sized to n it is $\mathcal{O}(n \log n)$). In contrast, SMT builds the tree bottom-up in a single pass over a known set ($\mathcal{O}(n)$ total hashes). Tree population happens off-circuit during construction, with both baselines passing pre-populated inputs to the same verification circuit, Algorithm 1. Our benchmark harness (§V) only alters the population routine.

2) Membership Proof and Verification: A membership proof for a secret s at leaf index i consists of the Merkle path $\{p_0, \ldots, p_{19}\}$ and directions $\{\delta_0, \ldots, \delta_{19}\}$. The verifier recomputes the path and checks it for consistency with the public root rt.

3) Circuit Complexity: Algorithm 1 performs 22 Poseidon invocations (one leaf hash, 20 path hashes, and one nullifier hash), compiling to **169 ACIR opcodes** and 9 Brillig opcodes, which Barretenberg expands into **1,795 gates**. IMT and SMT share this circuit exactly, so this cost is identical for both.

D. PEB-*A: Optimized Polynomial Equation Batching*

PEB-A is the optimized variant. It gives large initialization speedups over both Merkle baselines (§V), at the cost of an explicit polynomial-confidentiality assumption (Assumption 1) and a slower proof generation time.

1) Batch Representation and Commitment: We partition members into batches of fixed capacity $b = 128$. For each batch $B = \{s_1, \ldots, s_m\}$ with $m \leq b$, we construct a polynomial whose roots are exactly the member secrets:

$$P(x) = \prod_{j=1}^{m} (x - s_j) = c_0 + c_1 x + \cdots + c_{m-1} x^{m-1} + x^m$$

$P(x)$ is monic, so $c_m = 1$. The coefficient vector $\mathbf{c} = (c_0, \ldots, c_b)$ has length $b + 1$, with $c_j = 0$ for $j > m$. This

Algorithm 2 Polynomial Membership Verification Circuit (PEB-A)

Require: Public: commitment h, nullifier ν, verifier key k_V
Require: Private: secret s, KYC flag `isKYCed`, coefficients $\mathbf{c} = (c_0, \ldots, c_{128})$

assert $Poseidon(c_0,\ \ldots,\ c_{128}) = h$ ▷ Step 1: Commitment check
$y \leftarrow 0$ ▷ Step 2: Initialize Horner accumulator
for $j \leftarrow 128$ **down to** 0 **do**
  $y \leftarrow y \cdot s + c_j$ ▷ Step 3: Horner evaluation (129 steps)
end for
assert $y = 0$ ▷ Step 4: Membership $P(s) = 0$
assert `isKYCed` $=$ `true` ▷ Step 5: KYC check
assert $Poseidon(s,\ k_V) = \nu$ ▷ Step 6: Nullifier check

padding does not alter the root set. The vector uniquely determines the root multiset. The circuit uses Horner's method to evaluate the polynomial, and the batch commitment is a single Poseidon hash of the vector:

$$h = Poseidon(c_0, \ldots, c_b)$$

The issuer publishes h on-chain while the coefficients remain in its secured infrastructure (Assumption 1).

2) Batch Assignment: For populations exceeding b, members are partitioned via a deterministic mapping $\mathsf{Assign} : ref \to \mathsf{id}_B$. The exact routing policy is an application-layer design choice, provided each batch holds at most b members. Our reference implementation adopts a straightforward FIFO queue, $\mathsf{Assign}(ref) = \lfloor \mathrm{index}(ref)/b \rfloor$, to guarantee full batch packing density. At proof time, the issuer retrieves $\mathsf{id}_B \leftarrow \mathsf{Assign}(ref)$ and then only that batch's coefficient vector in $\mathcal{O}(1)$ time.

At $n = 2^{20}$ with $b = 128$, Assign yields 8,192 independent batches. Each publishes a single 32-byte commitment (one field element in $\mathbb{F}_p$), giving an $\mathcal{O}(n/b)$ on-chain footprint of 256 KiB ($8{,}192 \times 32$ B). A single depth-20 Merkle tree needs only one 32-byte root for the same population, and a multi-tree deployment has an $\mathcal{O}(n/2^{20})$ root count. PEB therefore trades a larger on-chain footprint for cheaper ingestion. We do not aggregate batch commitments in this work. The parameter b can be adjusted to balance on-chain storage with cost per proof. Increasing b linearly reduces the number of batches but proportionally increases the evaluation time of each proof (Algorithm 2, Step 3), since evaluation time scales as $\mathcal{O}(b)$. We set $b = 128$ as an operating point and the characterization of this trade-off is future work (§VIII-A).

3) Membership Proof and Verification: A membership proof for a secret s in the batch identified by id_B uses the private coefficient vector $\mathbf{c}$ and the public commitment h. Verification performs a commitment check, a Horner-evaluation membership check, and a nullifier check (Algorithm 2).

4) Security Assumption and Theorem: PEB-A introduces the algebraic threats analyzed in Section IV-H. It targets the Issuer-as-Prover setting and keeps polynomial state inside the issuer boundary. Protocol-level security (Definitions 1–3) follows

from cryptographic primitives. Assumption 1 formalizes the internal witness-at-rest confidentiality.

Assumption 1 (Private Polynomial Assumption)**.** *The issuer retains the coefficient vector $(c_0, \ldots, c_b)$ as internal private state at rest, disclosing* zero *coefficients outside its trust boundary. Only commitment h and verified proof transcripts π are observable externally.*

Theorem 1 (Security of PEB-A)**.** *Under simulation-extractability and zero-knowledge of the underlying zk-SNARK, and collision resistance of Poseidon,* PEB*-A satisfies Definitions 1, 2, and 3. Under Assumption 1, the coefficient vector is additionally never exposed outside the issuer's trust boundary, independent of these protocol-level guarantees.*

Proof. **Correctness.** By construction, the secret s of a member is a root of $P(x)$, and Horner evaluation yields $P(s) = 0$, as verified directly in-circuit.

Soundness. An adaptive adversary issuing at most q proof queries must either (i) distinguish q simulated proof responses with advantage at most $q \cdot \varepsilon_{\mathsf{ZK}}$, (ii) fail witness extraction under simulation-extractability with probability at most $\varepsilon_{\mathsf{SE}}$ [16], or (iii) extract a witness whose coefficient vector $\mathbf{c}$ differs from the honest vector while satisfying $Poseidon(\mathbf{c}) = h$. The error terms are defined in Appendix A. Case (iii) yields a direct Poseidon collision, occurring with probability at most $\varepsilon_{\mathsf{Poseidon}}$. Once $\mathbf{c}$ is pinned to the honest vector, $P(x)$ is fixed and Step 4's Horner evaluation enforces membership deterministically, leaving no further forgery path. Summing these failure events bounds the forgery probability by $q \cdot \varepsilon_{\mathsf{ZK}} + \varepsilon_{\mathsf{SE}} + \varepsilon_{\mathsf{Poseidon}}$ (full hybrid argument in Appendix A-C).

Zero-knowledge. The simulator $\mathsf{Sim}(h, \nu, k_V)$ invokes $\mathsf{Sim}_{\mathsf{SNARK}}$ on the public statement (h, ν, k_V) without private witness access. Transcript indistinguishability follows directly from the zero-knowledge property of the SNARK [15]. This argument applies identically to PEB-B (full simulator construction in Appendix A-B).

Witness confidentiality. Under Assumption 1 and the formal threat model (§III-D), the issuer discloses zero coefficients outside its trust boundary. An adversary observes only public parameters including h, nullifier ν, and proof π. Recovering the private witness $(\mathbf{c}, s)$ from this view requires either inverting Poseidon (to extract $\mathbf{c}$ from h or s from ν) or breaking SNARK soundness to extract a witness from π. This matches the primitives and bounds established above. This guarantee therefore stems directly from Assumption 1's disclosure restriction and not from a standalone reduction. □

E. PEB*-B: Hardened Equation Batching*

Standard Security Model: no assumptions beyond collision resistance of Poseidon and simulation-extractability and zero-knowledge of the zk-SNARK.

Algorithm 3 Polynomial Membership Verification Circuit (PEB-B)

Require: Public: commitment h, nullifier ν, verifier key k_V
Require: Private: secret s, KYC flag `isKYCed`, coefficients $\mathbf{c} = (c_0, \ldots, c_{128})$

1: **assert** $Poseidon(c_0, \ldots, c_{128}) = h$ ▷ Step 1: Commitment check
2: $\rho \leftarrow Poseidon(s)$ ▷ Step 2: Hash secret to root
3: $y \leftarrow 0$ ▷ Step 3: Initialize Horner accumulator
4: **for** $j \leftarrow 128$ **down to** 0 **do**
5: $\quad y \leftarrow y \cdot \rho + c_j$ ▷ Step 4: Horner evaluation at ρ
6: **end for**
7: **assert** $y = 0$ ▷ Step 5: Membership $P(\rho) = 0$
8: **assert** `isKYCed` $=$ `true` ▷ Step 6: KYC check
9: **assert** $Poseidon(s, k_V) = \nu$ ▷ Step 7: Nullifier check

PEB-B makes a single modification to PEB-A. Each member secret is hashed before it becomes a polynomial root:

$$s_i \to \rho_i = Poseidon(s_i), \qquad P(x) = \prod_{i=1}^{m} (x - \rho_i).$$

The issuer computes ρ_i off-circuit when it adds a member. The circuit hashes the secret s before the Horner evaluation (Algorithm 3). With roots $\rho_i = Poseidon(s_i)$, the coefficient vector $\mathbf{c}$ encodes preimage-resistant digest roots. Public exposure of $\mathbf{c}$ therefore reveals only the set $\{\rho_i\}_{i=1}^{m}$. For high-entropy secrets, recovering any s_i from it requires breaking the preimage resistance of Poseidon. Therefore, the scheme remains secure for high-entropy secrets even if $\mathbf{c}$ is leaked.

Theorem 2 (Security of PEB-B)**.** *Under simulation-extractability and zero-knowledge of the underlying zk-SNARK, and collision resistance of Poseidon,* PEB*-B satisfies Definitions 1, 2, and 3.*

Proof. **Correctness.** Identical to Theorem 1, with the roots s_i replaced by hashed roots $\rho_i = Poseidon(s_i)$ such that $P(\rho_i) = 0$.

Soundness. An adaptive adversary issuing at most q queries succeeds via cases (i)–(iii) from Theorem 1, or via case (iv). Case (iv) supplies $s^* \neq s_k$ with $Poseidon(s^*) = \rho_k$ for some member s_k of the batch committed in h^*. In the soundness experiment the reduction plays the issuer and knows every member secret, so (s^*, s_k) is directly a collision for Poseidon. Case (iv) therefore occurs with probability at most $\varepsilon_{\mathsf{Poseidon}}$. Summing failure probabilities across all cases bounds the overall forgery probability by $q \cdot \varepsilon_{\mathsf{ZK}} + \varepsilon_{\mathsf{SE}} + 2\varepsilon_{\mathsf{Poseidon}}$ (full hybrid argument in Appendix A-C).

Zero-knowledge. The argument is identical to Theorem 1. The simulator construction and zero-knowledge bounds carry over directly to PEB-B (Appendix A-B). □

F. External Protocol

Definition 4 (Polynomial Accumulator Scheme)**.** *A Polynomial Accumulator Scheme* Π *refines the generic accumulator*

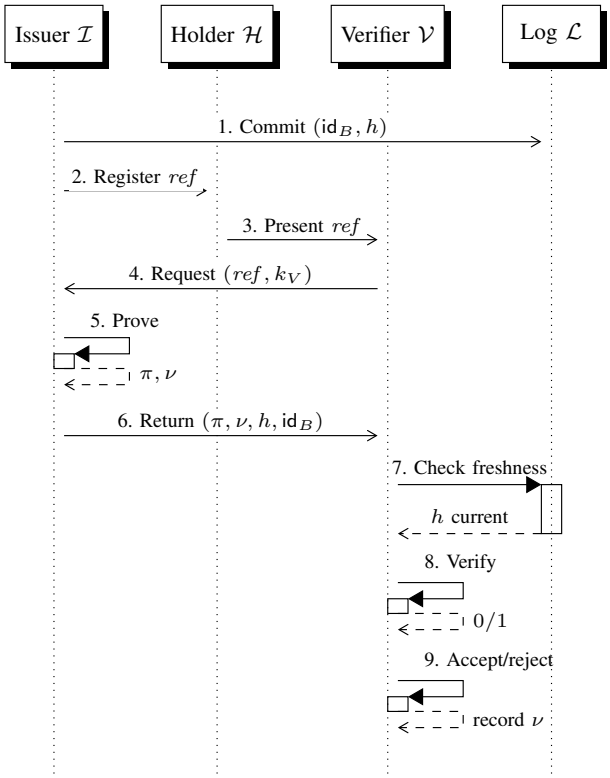


Fig. 2: External protocol for PEB-A/B.

syntax into five polynomial-time algorithms, decoupling local accumulation from public commitment:

- $\mathsf{Setup}(1^\lambda, b) \to (pk, vk)$*: on security parameter* λ *and batch capacity* b*, outputs proving key* pk *and verification key* vk*. Run once at deployment.*
- $\mathsf{Add}(\mathbf{c}, a) \to \mathbf{c}'$*: given the coefficient vector* $\mathbf{c}$ *of* $P(X) \in \mathbb{F}_p[X]$ *with* $\deg(P) < b$*, extends* P *by a new root* a *via* $P'(X) = P(X)(X - a)$ *and returns* $\mathbf{c}'$*, with* $\deg(P') = \deg(P) + 1$*. The root* a *is the member secret* s *in* PEB-*A and* $Poseidon(s)$ *in* PEB-*B.* Add *is defined only while* $\deg(P) < b$*. Iterating from* $\mathbf{c}^{\text{init}} = (1, 0, \ldots, 0)$ *supports incremental or batch-wise construction up to capacity* b*.*
- $\mathsf{Commit}(\mathbf{c}, \mathsf{id}_B) \to h$*: given* $\mathbf{c}$ *encoding* $P(X) \in \mathbb{F}_p[X]$ *with* $\deg(P) \le b$ *and batch id* id_B*, zero-pad* $\mathbf{c}$ *to length* $b + 1$ *(if needed) and output* $h = Poseidon(\mathbf{c})$ *via an authenticated* `BatchCommitted` *event carrying* id_B*.* Commit *may be called multiple times per batch; the latest* h *overrides earlier ones.*
- $\mathsf{GenProof}(pk, x_{pub}, x_{priv}) \to \pi$*: using proving key* pk*, public input* x_{pub}*, and witness* x_{priv}*, output* π *proving that the root derived from secret* $s \in x_{priv}$ *(*s *for* PEB-*A,* $Poseidon(s)$ *for* PEB-*B) is a root of the polynomial encoded by* $\mathbf{c} \in x_{priv}$ *and committed as* $h \in x_{pub}$*.*
- $\mathsf{Verify}(vk, x_{pub}, \pi) \to \{0, 1\}$*: check* π *against* vk *and* x_{pub}*.*

Definitions 1–3 are stated for the generic syntax of §IV-B. They instantiate to Π as follows. The commitment C is h, the public instance is $x_{\text{pub}} = (h, \nu, k_V)$, and the private witness is $x_{\text{priv}} = (s, \mathbf{c}, \mathsf{isKYCed})$. The generic $\mathsf{Commit}(S)$ is the composition of Add over the elements of S followed by $\mathsf{Commit}(\mathbf{c}, \mathsf{id}_B)$. Stating the definitions generically lets Appendix A reuse them for the IMT/SMT baselines (Definitions 5–8), whose algorithms differ from Π's.

Parties. Figure 2 involves an Issuer $\mathcal{I}$ (builds and custodies each batch), a Holder $\mathcal{H}$ (a registered member), a Verifier $\mathcal{V}$, and a public Event Log $\mathcal{L}$ of `BatchCommitted` events. Per Design Goal P3, $\mathcal{I}$ is the sole prover, so $\mathbf{c}$ and member secrets never leave its infrastructure. Each verifier $\mathcal{V}$ holds a key k_V registered with $\mathcal{I}$.

Protocol.

1) **Commit:** $\mathcal{I}$ accumulates $\mathbf{c}$ off-chain via repeated Add calls and posts (id_B, h) with $h \leftarrow \mathsf{Commit}(\mathbf{c}, \mathsf{id}_B)$ to $\mathcal{L}$. Since Commit may run more than once per batch, $\mathcal{L}$ can hold several historical h for the same id_B.
2) **Register:** $\mathcal{I} \to \mathcal{H}$: a membership reference ref (e.g., an account or credential ID) that determines id_B. s and $\mathbf{c}$ stay with $\mathcal{I}$.
3) **Present:** $\mathcal{H} \to \mathcal{V}$: ref, to prove membership.
4) **Request:** $\mathcal{V} \to \mathcal{I}$: (ref, k_V). $\mathcal{I}$ authenticates $\mathcal{V}$ and accepts k_V only if it is the key registered to $\mathcal{V}$. k_V scopes the nullifier (Step 5) to this issuer-verifier pair.
5) **Prove:** $\mathcal{I}$ resolves ref to $(s, \mathbf{c}, \mathsf{id}_B)$ and sets $\nu \leftarrow Poseidon(s, k_V)$, $x_{\text{pub}} \leftarrow (h, \nu, k_V)$, and $x_{\text{priv}} \leftarrow (s, \mathbf{c}, \mathsf{isKYCed})$. It then runs $\pi \leftarrow \mathsf{GenProof}(pk, x_{\text{pub}}, x_{\text{priv}})$ under Algorithm 2 (PEB-A) or Algorithm 3 (PEB-B). The circuit enforces the commitment, Horner-membership, KYC, and nullifier checks.
6) **Return:** $\mathcal{I} \to \mathcal{V}$: $(\pi, \nu, h, \mathsf{id}_B)$. If s has sufficient min-entropy, neither s nor $\mathbf{c}$ is computationally recoverable from this tuple.
7) **Check freshness:** $\mathcal{V}$ confirms h matches the latest `BatchCommitted` value posted by $\mathcal{I}$ for id_B on $\mathcal{L}$, rejecting any outdated h before SNARK verification.
8) **Verify:** $\mathcal{V}$ sets $x_{\text{pub}} = (h, \nu, k_V)$ and evaluates $\mathsf{Verify}(vk, x_{\text{pub}}, \pi) \to \{0, 1\}$, typically via an on-chain Verifier Contract with vk hardcoded. The check is unchanged off-chain (e.g., TEE-backed or permissioned deployments).
9) **Accept/reject:** $\mathcal{V}$ accepts iff Steps 7 and 8 both pass, then records ν. Otherwise $\mathcal{V}$ rejects.

Authorization is issuer-bound. Soundness (Definition 2) only ensures that h commits to some valid coefficient vector, not that it is the authorized batch. A prover could instead commit to a polynomial rooted in its own secret. Step 7 binds h to$\mathcal{I}$ posted to $\mathcal{L}$, and $\mathcal{L}$ authenticates the poster, so it rejects a proof against an unauthorized or superseded commitment. The tuple (h, ν, k_V) still links a proof only to a legitimate batch member, not unique to the Holder. That binding and the private input isKYCed, therefore, relies on the honesty of $\mathcal{I}$ (P3), not Definitions 1–3. Step 4 also prevents third parties from eliciting proofs under another verifier's key to obtain that verifier's nullifier or a replayable tuple.

Nullifiers. The nullifier $\nu = Poseidon(s, k_V)$ provides (i) cross-verifier unlinkability, since under Poseidon's keyed indistinguishability (§III-D) and zero-knowledge, the proofs and nullifiers received by distinct verifiers cannot be linked to

each other or to s, and (ii) within-verifier detection of repeated ν. The unlinkability rests on keyed indistinguishability and is not part of Theorems 1–3. It covers only the proof and nullifier. Colluding verifiers can still link a holder through a shared ref unless it is scoped per verifier, and (h, id_B) narrows the holder to one batch of at most b members. Unlinkability also does not hold against $\mathcal{I}$, which sees every (ref, k_V) request. In our setting $\mathcal{I}$ already holds the member's KYC record, so it learns only which verifiers the member contacts. Because $\mathcal{V}$ initiates each request, a repeated ν usually signals re-verification rather than replay. Whether to reject, rate-limit, or accept it is an application-layer choice, as in prior nullifier-based systems [22], [23], [24]. Nullifiers do not expire on a status change, and nullifier-aware revocation is left to future work (§VIII-A).

Self-sovereign variant. $\mathcal{I}$ may instead distribute $\mathbf{c}$ so each Holder runs Step 5 independently, weakening the threat model of §III-D. Appendix A-E analyzes this non-default variant for PEB-B.

G. Complexity Comparison

TABLE I: Computational and circuit complexity (n = total members, b = batch size = 128).

Operation / Metric	IMT	SMT	PEB
Single insertion (amortized)	$O(\log n)$ §	–	$O(1)$ w.r.t. n †
Batch/tree initialization (n mem.)	$O(n \log n)$ §	$O(n)$	$O(n)$ †
Verification complexity	$O(\log n)$ §	$O(\log n)$ §	$O(1)$ w.r.t. n †
Circuit size (ACIR opcodes)	169 (shared)		304 / 306 ‡
Circuit size (UltraHonk gates)	1,795 (shared)		3,655 / 3,729 ‡
Additional-capacity strategy	additional tree		additional batch

† Each batch costs $O(b^2)$ field operations via sequential polynomial multiplication, so $(n/b) \times b^2 = O(nb)$. With fixed $b = 128$, both $O(b)$ and $O(nb)$ reduce to $O(1)$ and $O(n)$ w.r.t. n respectively. IMT and SMT share an identical proof circuit and differ only in population strategy.

§ For a tree sized to n. Our fixed depth of 20 gives 21 hashes per insertion, a total cost linear in n, and a constant 22-hash verification circuit.

‡ PEB-A / PEB-B, respectively.

The practical gap exceeds the asymptotic ratio. The Merkle baselines are dominated by native Poseidon evaluations, whereas PEB relies primarily on field multiplications. Evaluating an arithmetization-oriented hash costs many field multiplications [25], [30]. The population advantage of PEB therefore exceeds what hash counts alone imply (see Appendix B-A).

Circuit overhead is bounded. PEB has about $2.0\times$ more UltraHonk gates than the Merkle circuit (3,655 for PEB-A and 3,729 for PEB-B, against 1,795) and about $1.8\times$ more ACIR opcodes. The extra gates come mainly from the commitment check. Because the gate overhead is fixed and independent of n or batch count, it is a constant proof-generation penalty relative to IMT and SMT, whether capacity is expanded via additional trees or batches.

H. Algebraic Attack Analysis

1) Threat: Full-Vector Coefficient Leakage: In PEB-A, member secrets are roots of $P(x)$ directly. If the coefficient vector is ever exposed, an adversary recovers *every* member secret in the batch by root finding over BN254 in polynomial time. A single leak is total compromise. IMT/SMT do not have this problem, since their leaves store $Poseidon(s_i)$.

2) Threat: Multi-Equation Polynomial Division: Any two disclosures of the same batch's polynomial are algebraically linked by their quotient or greatest common divisor (GCD), whether they come from a before-and-after removal pair or from separate storage breaches. A single insertion or removal makes the quotient equal to $x-a$ or its inverse, where a is the changed root. The general case is resolved via $\gcd(P_1(x), P_2(x))$. Under PEB-A this recovers the changed secret directly, so total compromise is now reachable from two historical leaks instead of one live leak.

3) Resolution and Trade-off: In PEB-B the roots are $Poseidon(s_i)$, so root recovery and polynomial division yield only hashed values. Linking one to a member requires reversing Poseidon, which is infeasible for high-entropy secrets (formalized in Appendix A-C). This guarantee is cryptographic and holds independent of custody, which PEB-B keeps only as defense-in-depth (Design Goal P3).

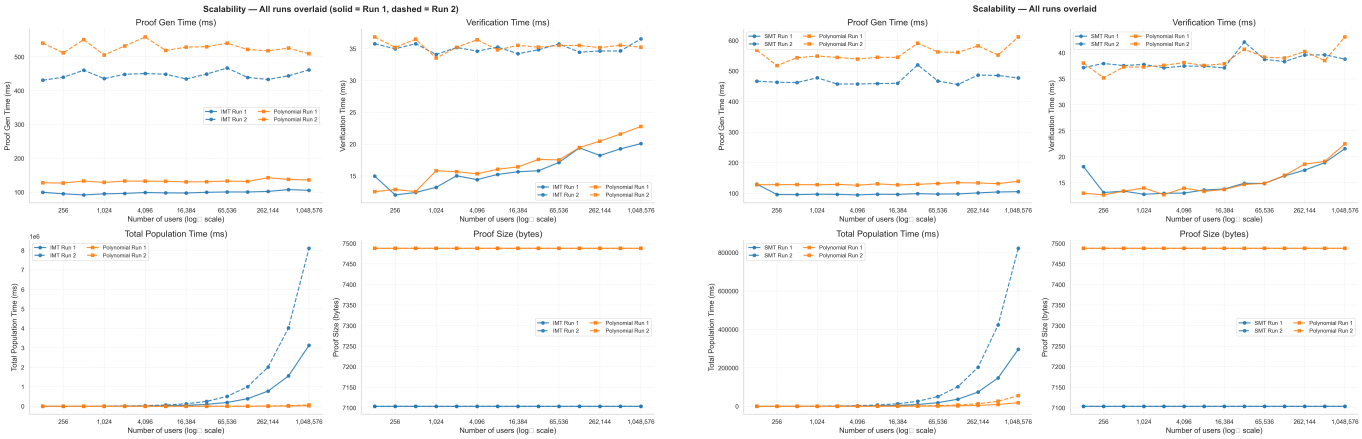
PEB-A has no such guarantee. Its defense is operational. The issuer alone holds the coefficient vector (Design Goal P3, Section IV-F), and only h is published, so an attack requires breaching this boundary, for example through unpurged logs, backups, or removal diffs. Operators should treat superseded or duplicate coefficient vectors as security-critical material.

The choice between the variants is therefore a custody question and not a security ranking. PEB-A trades a stronger operational dependence (Assumption 1) for speed, which fits deployments where that trust boundary already holds, such as TEE-backed or permissioned settings. PEB-B pays the modest cost quantified in §V to remove that dependence. Both keep substantial initialization speedups over IMT and SMT.

Theorem 3 (Security of PEB)**.** *Under the collision and preimage resistance of Poseidon, the simulation-extractability and zero-knowledge of the zk-SNARK, and the custody model of Section IV-F (plus Assumption 1 for* PEB-*A), both* PEB-*A and* PEB-*B satisfy Correctness, Soundness, and Zero-Knowledge (Theorems 1–2). They also satisfy state binding, append-only consistency, and public verifiability, and revocation feasibility in principle (Appendix A).*

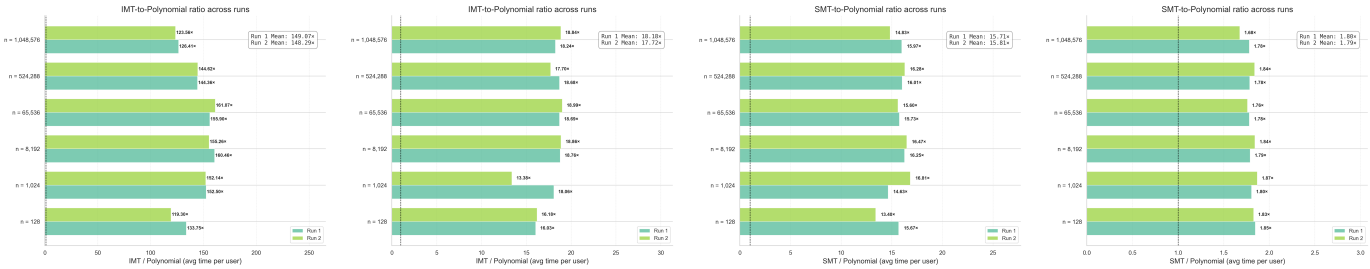
V. Evaluation

We characterize PEB's own scaling behavior and then situate it against two operationally distinct Merkle tree baselines: an incrementally updated tree (IMT) and a bulk, bottom-up static tree (SMT) built from a known member set in advance. Both baselines share the same membership-proof circuit and differ only in their population strategies, thereby isolating the comparison to the relevant axes: initialization cost and proof-generation cost. Each PEB variant is compared against *both* baselines, giving four pairwise comparisons in total. We benchmark both the optimized PEB-A and the hardened PEB-B across 14 member counts ranging from 2^7 to 2^{20} users.



(a) IMT vs. PEB-A (incremental) (b) SMT vs. PEB-A (bulk)

Fig. 3: Scalability comparison across 14 population sizes for PEB-A across Run 1 (solid) and Run 2 (dashed). Clockwise from top-left: proof generation time, verification time, proof size, total population time (log-log).



(a) IMT vs. PEB-A (b) IMT vs. PEB-B (c) SMT vs. PEB-A (d) SMT vs. PEB-B

Fig. 4: Per-user speedup ratios across all four pairings for Run 1 and Run 2. The bars show six representative population sizes. The legend in each panel reports the mean over all 14 sizes.

A. Experimental Setup

1) Implementation: All constructions use Noir version 1.0.0-beta.15 [11] and compile to ACIR, relying on Barretenberg version 3.0.0-nightly.20251104 [12] for proving. All variants share the same proving infrastructure. Only their population routines and circuit constraints differ. Off-circuit hashing uses `zkpassport/poseidon2`, while PEB uses custom TypeScript field arithmetic for polynomial construction, depending on each baseline's population strategy.

TABLE II: Benchmark run configurations.

Run	CPU	RAM	OS
1 (primary)	AMD Ryzen 7 7840U, 8-core	32 GB	Ubuntu 24.04.4 (native)
2 (repro.)	AMD Ryzen 5 7530U, 6-core	32 GB	Ubuntu 24.04.1 (WSL2)

2) Benchmark Runs: Table II summarizes our two benchmark environments: Run 1 (primary) and Run 2 (reproducibility). Both runs cover all four pairings across the full range of 14 population sizes, enabling cross-environment validation. Executions are strictly single-threaded to isolate algorithmic performance from parallelization effects, utilizing the Noir and Barretenberg versions noted previously.

3) Methodology: Each Merkle baseline is compared against PEB using its native population strategy: incremental for IMT and bulk for SMT. We evaluate four independent pairings (IMT/PEB-A, IMT/PEB-B, SMT/PEB-A, and SMT/PEB-B) across both benchmark runs (Table III). **All reported summary speedups and relative overheads are within-pairing metrics computed as the arithmetic mean of per-size ratios across all 14 population sizes.**

For each population size n and pairing, we measure: (i) cumulative and per-user population time, (ii) averaged proof generation and verification time over five random members, (iii) accumulator storage size, and (iv) proof and verification-key sizes. Verification time refers to native SNARK verification (`bb verify`) execution. Gas cost is out of scope.

We maintain complete in-memory tree structures for IMT and SMT to enable direct path lookups. While production Merkle systems often delegate path retention to clients, PEB requires the issuer to retain the coefficient vector for proof generation (§III). Consequently, our storage and population benchmarks represent a conservative evaluation, comparing PEB's mandatory full-retention footprint against each Merkle baseline's best-case latency configuration.

Because circuit constraint counts are fixed and input-

independent, we report single-run wall-clock times for structural metrics following standard conventions [31], [32]. Minor timing variations are attributed to host-level scheduling noise, with comprehensive multi-trial variance constrained by benchmarking resources (§VIII-A).

B. Results

1) Cross-Environment Reproducibility: Table III reports population speedups, raw mean proof-generation times ($T_{\text{base}}/T_{\text{PEB}}$), and relative overheads ($\Delta T/T_{\text{base}}$) across Run 1 (native Linux) and Run 2 (WSL2). Speedup and overhead metrics are reported as the mean of per-size ratios across all 14 population sizes. Speedups match within 3% across environments ($148.29\times$–$149.07\times$ for IMT/A; $1.79\times$–$1.80\times$ for SMT/B). This suggests that the speedup does not depend strongly on the host.

However, the relative overhead decreased from 30.11%–33.87% in Run 1 to 18.45%–20.41% in Run 2. Run 2 operates on weaker hardware than Run 1, with the WSL2 virtualization layer adding secondary overhead (Table II). Across all four pairings, baseline proof generation slows by $4.50\times$–$5.13\times$ under this host change, while the additional overhead (ΔT) grows by only $2.46\times$–$2.96\times$. The baseline denominator (T_{base}) therefore grows faster than the delta numerator (ΔT), which lowers the ratio $\Delta T/T_{\text{base}}$.

We attribute this to a fixed cost in measured proof-generation time that is independent of circuit size, like process start-up, key loading, and file I/O. This cost is shared by the baseline and PEB and is likely inflated under WSL2. The extra work of PEB is compute-bound and scales mainly with the CPU gap. This also explains why Run 1 is faster overall. Verifying this requires a native Linux host with Run 2 hardware and is outside the scope of this paper. Our conclusions rest on the measured overheads and do not depend on this explanation.

TABLE III: Cross-environment reproducibility across pairings.

	Mean Proof Gen Time: Base / PEB (ms)		Speedup		Overhead	
Pairing	**R1**	**R2**	**R1**	**R2**	**R1**	**R2**
IMT/A	99.11 / 132.56	446.06 / 528.18	149.07×	148.29×	33.87%	18.46%
IMT/B	98.59 / 128.14	505.90 / 593.26	18.18×	17.72×	30.11%	20.41%
SMT/A	100.82 / 130.75	471.36 / 558.11	15.71×	15.81×	30.44%	18.45%
SMT/B	95.51 / 125.72	473.58 / 561.52	1.80×	1.79×	31.68%	18.70%

2) Incremental Insertion: IMT *vs.* PEB*:* Figure 3a compares IMT with PEB-A under incremental population. The absolute gap in population time grows with n in both environments. This speedup reflects the native, off-circuit cost of a Poseidon evaluation versus a field multiplication (§III and Appendix B-A). Table III lists the per-pairing speedups and proof-generation overheads. PEB-B has a lower speedup than PEB-A, yet it outperforms IMT in both environments.

3) Bulk Insertion: SMT *vs.* PEB*:* Figure 3b compares bulk-insertion performance of SMT and PEB-A. SMT has proven as a strong baseline, achieving mean per-user population speedups of $11.04\times$ in Run 1 and $9.81\times$ in Run 2 for the PEB-A pairing blocks, and $10.61\times$ and $10.65\times$ for the PEB-B blocks. However, Table III shows that PEB-A still achieves substantial speedups over SMT, while PEB-B keeps a smaller lead.

4) Proof Size, Verification and Storage: The verifier-side and storage costs of PEB are modest. Its proofs are 7,488 B versus 7,104 B for Merkle baselines, and each structure has a 1,888 B verification key. The 384 B gap in proof size comes from the circuit size. UltraHonk pads each circuit to a power of two, and proof size grows logarithmically with the padded size. The Merkle circuit has 1,795 gates and pads to 2^{11}, while the PEB circuits have 3,655 (A) and 3,729 (B) gates and both pad to 2^{12} (Table I). They therefore sit exactly one doubling higher, which also explains why the two variants share the same proof size. Verification time is similar across structures in both runs. In Run 1 it rises from about 12 ms at small n to about 23 ms at $n = 2^{20}$, and in Run 2 it stays between about 35 and 45 ms. Storage is equally close. At $n = 2^{20}$ in Run 1, PEB occupies 176.06 MB and IMT and SMT occupy 180.81 MB.

C. Summary

Figure 4 shows the speedups per user for the population across all four comparisons in both benchmark runs. The speedups are similar across the two environments. Both variants of PEB have lower population time than both tree baselines in their respective optimal insertion strategies. The gains against the incremental IMT are substantial. The margins narrow when comparing against the bulk-optimized SMT, particularly for PEB-B. PEB nonetheless remains faster in population time in all four pairings. This population advantage comes with a trade-off in proof generation speed (Table III). Verification times are similar across structures in both environments (§V-B4). Here, PEB-B trades speed over hardened security by addressing the algebraic threats discussed in §IV-H. Consequently, it has a higher initialization cost than PEB-A, despite a comparable proof-generation overhead. These results clarify the trade-off under Axis 2 (§III). PEB is preferable when population cost dominates and a custodial issuer-as-prover model is acceptable. Merkle trees remain preferable when proof-generation latency is the primary bottleneck.

VI. Discussion

The trade-off established in §III-A maps to two deployment contexts. Both involve a custodial issuer and not self-sovereign proving. Our pairwise evaluation of IMT, bulk SMT, and both PEB variants also answers the call of NIST's Privacy-Enhancing Cryptography project for benchmarked proofs of concept over abstract claims [33]. Recent SoK work on SNARK deployment identifies the same gap [31], [32].

The first context is private writes. The Ethereum Foundation's Privacy Stewards roadmap [34] treats them as distinct from on-device proving. PEB's initialization advantage (Axis 1) targets the write side, and we leave on-device proving aside as an orthogonal track. The proof-generation overhead is about 30% to 34% in Run 1 and roughly constant across population sizes (§V), so the initialization gain outweighs it when few proofs are generated per member. In proof-dominated systems

such as ZK-rollup sequencing or high-throughput financial settlement [3], the overhead is hard to justify, and Merkle accumulators remain the better choice. PEB also has a larger on-chain footprint than a single Merkle root (§IV-D).

The second context is compliance interoperability. The EU's eIDAS 2.0 wallet regulation [5] and MiCA's compliance obligations [7] create demand for a chain-agnostic layer. In the flow we envision, a regulated entity verifies a user once through European Digital Identity Wallet (EUDIW) flows and FATF-aligned due diligence [4], screens sanctions lists [26], [27], and adds the user to a batch accumulator. It then generates compliance proofs on the user's behalf across chains without re-verification. This is issuer-side re-proving under Axis 2.

In both contexts, an issuer, exchange, or regulated entity is the natural custodian of the membership set. The same model may extend to other onboarding-dominant applications, such as supply-chain provenance and enterprise credentialing, etc. When issuer custody is acceptable, both PEB variants initialize faster than both Merkle baselines in our benchmarks. Against incremental IMT, PEB-A has a mean initialization speedup of $149.07\times$ in Run 1, at a mean proof-generation overhead of 33.87%. This suits systems where the operator-confidentiality boundary of Assumption 1 is already set by the deployment context (P3), such as TEE-backed services, permissioned DAOs, or regulated VASP infrastructures. Against a bulk-optimized SMT, the margin narrows. However, PEB-B still initializes $1.80\times$ faster in Run 1 and adds hashed-root protection against algebraic threats (§IV-H), which makes it the relevant comparison for deployments that require hashed roots. Appendix B-A analyzes the drivers of this margin.

The issuer-as-prover model also has systems-level costs beyond the guarantees of Theorem 3. The issuer must stay online for every verification, not just during onboarding, and it observes every verification request (§IV-F). Deployed IMT-based systems avoid this single point of failure once paths are distributed locally [22], [23], [24]. A PEB deployment keeps a centralized dependency as long as the issuer exclusively holds the coefficient vector. This creates availability and censorship risks that are separate from correctness, soundness, and zero-knowledge (Definitions 1–3). Designers should therefore check that their domain satisfies Axis 2's applicability gate (§III) before choosing PEB over a Merkle-tree baseline.

VII. Related Work

Cryptographic accumulators for zero-knowledge membership have evolved to overcome key trade-offs in witness size, update cost, and circuit complexity. Table IV compares major constructions across these dimensions alongside PEB. We trace this progression below.

Tree-based accumulators [21], [35] offer $\mathcal{O}(\log n)$ proof sizes, but each insertion requires iterative updates and forces members to manage their own authentication paths. Merkle trees are widely used in proof-frequent privacy systems, including Zcash's shielded pool [36], Semaphore [22], Tornado Cash [23], and Privacy Pools [24]. Verkle trees [37], [38] compress proofs through vector commitments, but maintenance overhead remains, and KZG-based instantiations require a trusted setup. Curve Trees [39] and Curve Forests [40] provide zero-knowledge set membership without a trusted setup, using cycle-of-curves constructions. Forests extend Curve Trees with batching and stronger security guarantees.

RSA accumulators [17], [19], [18], [41] achieve $\mathcal{O}(1)$ proof sizes but typically require a trusted setup and expensive in-circuit group arithmetic. Recent works such as HARISA [42] and Benarroch et al. [43] optimize proving times for simultaneous multi-element proofs. Constructing the accumulator still demands group exponentiations, whereas PEB relies on finite-field operations.

Bilinear accumulators [20], [44] extend constant-size membership and non-membership proofs to pairing-friendly groups under q-type strong Diffie-Hellman assumptions. Srinivasan et al. [8] improve batching and zero-knowledge proof aggregation in this setting while keeping the underlying trusted setup and group-exponentiation costs.

Polynomial & lookup approaches share PEB's algebraic core. Bayer and Groth [45] introduced root-encoding polynomials ($P(u) = 0$) for zero-knowledge blacklist membership. They commit via pairings and prove membership of one element at a time under a trusted setup. PEB adapts root encoding to a batch-initialized setting without a trusted setup in the accumulator layer. It allows a bounded batch of up to 128 members under a single Poseidon commitment. Membership is verified through in-circuit Horner evaluation and not through a pairing check. KZG-based constructions [46] provide $\mathcal{O}(1)$ witnesses but require a trusted setup and pairings. Lookup arguments [47], [48] target fixed preprocessed tables, unlike PEB's fresh per-epoch Poseidon commitments.

PEB occupies a distinct position among these families. Its accumulator layer has no trusted setup, and it is batch-initialized algebraically and does not rely on incremental updates, tree hashing, group exponentiation, or preprocessed lookup structures. Its onboarding cost is dominated by finite-field multiplication, with native support for per-epoch refresh. To our knowledge, no polynomial-based accumulator achieves this onboarding profile and also supports isolated, self-sovereign membership witnesses without pairing-based verification or a trusted or updatable setup. Constructions that combine these properties, potentially through updatable polynomial commitments or hybrid tree-polynomial designs, remain an open research direction (§VIII-A).

VIII. Conclusion

We presented PEB, a polynomial-based accumulator for zero-knowledge membership verification in the onboarding-dominant regime, where initialization cost outweighs proof-generation frequency. PEB encodes batches of $b = 128$ members as roots of a characteristic polynomial and commits with a single Poseidon hash. It trades higher proof-generation cost and a larger on-chain footprint for lower initialization cost. The initialization gain reflects the native, off-circuit cost of a hash evaluation versus a field multiplication (Appendix B).

TABLE IV: Comparison of accumulator families for zero-knowledge membership proofs under performance, security, and system-model dimensions.

Criteria	**Merkle / Verkle Tree [21], [35], [37], [38]**	**RSA Acc. [19], [18]**	**Bilinear [20], [44]**	**Poly/Lookup [45], [47]**	**Curve-Tree [39], [40]**	**Our Work (PEB)**
Performance & Complexity[‡]						
Insertion Cost	$O(\log n)$ hashes[#]	1 modular exp.	1 group exp.	$O(d)$ mult.[§]	$O(w \log_w n)$ group ops[¶]	$O(b)$ field muls ($b = 128$)
Batch Init. Cost[†]	$O(n \log n)$ hashes (stream)	$O(n)$ modular exps	$O(n)$ group exps	$O(N \log N)$ FFT commit[‖]	$O(n)$ group ops[¶]	$O(n)$ field ops
Native Verification Complexity[*]	$O(\log n)$ hashes[#]	$O(1)$ exponentiation	$O(1)$ pairing check	$O(1)$ pairing check	$O(w \log_w n)$ field ops[‡‡]	$O(b)$ field ops, b fixed
System Model & Operations						
Batch / Snapshot Support	Possible but costly	Supported	Supported	Strong, committed table	Strong, batched membership	Strong, core design goal
Witness Dist. Model	Merkle path per user	Witness + public update info	Witness + public update info	Precomputed lookup witness	Per-batch commitment handles	Issuer-custodial (coeff. vector)
Security & Assumptions						
Membership Soundness	Hash collision resistance / Discrete Logarithm[#]	Strong RSA soundness	Pairing-based soundness	Polynomial lookup soundness	Discrete Logarithm	Hash collision resistance + SNARK soundness
Data Confidentiality	Good if leaves are hashed	ZK hides member witness	ZK hides member witness	ZK / position hiding	ZK membership privacy	ZK (private coeffs / hashed roots)

Families: tree-based (Merkle [21], [35] and Verkle [37], [38]), RSA [19], [18], bilinear [20], [44], polynomial (root-based [45] and lookup-based [47]), curve-tree [39], [40], and PEB (this work).

[‡] Rows in this block report asymptotic order only. Several families share $O(n)$ for batch initialization, so this block should not be read as showing PEB's advantage. That advantage is a constant-factor effect (field multiplication versus hash or group-exponentiation cost per element), reported empirically in Section V. It is not an asymptotic separation.

[†] The IMT figure is for streaming construction. A bulk-constructed static tree (SMT) achieves $O(n)$, see Table I.

[§] The two sub-families in this column differ operationally. Root-based polynomial evaluation [45] appends an element via one polynomial multiplication by $(x - r)$. This costs $O(d)$ field multiplications per insertion, where d is the current polynomial degree. This is the same cost model as PEB's own Insertion Cost row and is not a constant-time operation. Lookup arguments [47] instead target a fixed, preprocessed table reused across queries, for which single-element insertion is not the optimized operation. The Batch Init. and Verification rows describe the lookup-argument variant, which is the sub-family most directly comparable to PEB's batch-committed design.

[‖] N denotes the preprocessed lookup table size. It is distinct from the population n elsewhere in this table, and the two are not directly comparable row to row.

[¶] Curve Trees scale with the branching factor w per level. In Curve Forests, sequential amortized cost can approach $O(1)$ updates under specific batch assumptions, but the structural baseline for a single insertion remains $O(w \log_w n)$.

[*] Complexity of each family's verification algorithm, assuming the verifier holds the necessary witness data in the clear. For RSA and bilinear accumulators, this is a separate, non-private mode of the same scheme. Polynomial/lookup and curve-tree constructions are proof systems by design and have no non-proof counterpart. Their entries describe the underlying algebraic check (pairing equation or compressed group-operation argument) and not a stripped-down alternative.

[‡‡] w is the tree's branching factor. Curve Trees require verifying Pedersen commitments of size w at each layer of the depth $\log_w n$. When wrapped in a ZK-SNARK or Bulletproofs-style argument, the verification proof size shrinks asymptotically to $O(\log_w n)$, but the underlying circuit or native verification relation scales with $w \log_w n$.

[#] Verkle trees replace hashes with vector commitments (relying on Discrete Logarithm or q-SDH assumptions), reducing proof size to $O(\log_k n)$ or $O(1)$, while retaining $O(\log n)$ update complexities.

Our two variants show a security-efficiency trade-off. In our primary environment (Run 1), PEB-A uses the Private Polynomial Assumption (Assumption 1) and achieves a $149.07\times$ mean speedup over IMT, with a 33.87% proof-generation overhead. PEB-B hashes secrets in-circuit, which removes the algebraic threats for high-entropy secrets. It achieves an $18.18\times$ speedup over IMT at a 30.11% overhead. Against bulk-initialized SMT baselines, the speedups are $15.71\times$ (30.44% overhead) for PEB-A and $1.80\times$ (31.68% overhead) for PEB-B. Verification times are similar across all structures. A second environment gave speedups within 3% of these values.

To conclude, PEB is preferable when onboarding cost dominates, and an issuer-as-prover custodial model fits the deployment's trust model (§III-A). Merkle trees remain better when proof-generation frequency dominates or self-sovereign proving is required.

A. Limitations and Future Work

- **Witness Custody & Prover Liveness:** Requiring the complete $O(b)$ coefficient vector as a witness makes proof generation issuer-custodial, creating an operational dependency on issuer liveness. Supporting self-sovereign proving through updatable polynomial commitments or hybrid structures remains future work.
- **Revocation:** Revoking one member changes the batch commitment h and invalidates proofs for the whole batch. We show that revocation is possible in principle (Appendix A-D), but we do not implement or evaluate it.
- **Hardware Scope & Multi-Trial Variance:** Benchmarking on specialized hardware (e.g., GPUs, FPGAs) and multi-threaded proving environments, as well as extensive multi-trial statistical variance analysis across repeated execution runs, are left to future work.
- **Fixed Batch Size:** Trade-offs among constraint costs, on-chain footprint, and proof-generation latency across alternative batch sizes remain open for deployment tuning.
- **Nullifier-State Decoupling:** The nullifier operates independently of batch state and does not expire on a status change. Synchronizing nullifier state with batch updates is left to future work.
- **Accumulator Baselines:** Benchmarking was limited to standard incremental and static Merkle trees. Direct comparisons against Merkle forests, Verkle trees [37], bilinear accumulators [8], and curve trees [39], [40] are future work.

- **Scope of Claims:** The throughput gains do not reduce circuit or developer complexity. A Noir circuit must still be written and audited. We take the need for zero-knowledge as given in each cited application and focus on the accumulator's throughput.

REFERENCES

[1] V. Buterin, J. Illum, M. Nadler, F. Schär, and A. Soleimani, "Blockchain privacy and regulatory compliance: Towards a practical equilibrium," *Blockchain: Research and Applications*, vol. 5, no. 1, p. 100176, 2024.

[2] M. Rosenberg, J. White, C. Garman, and I. Miers, "zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure," in *IEEE Symposium on Security and Privacy (S&P 2023)*, 2023, pp. 790–808.

[3] T. Adrian, Y. Cabedo, and T. Mancini-Griffoli, "The rise of tokenization: Deciphering new trends in payments and asset tokenization," International Monetary Fund, Washington, DC, Tech. Rep. NOTE/2026/006, 2026. [Online]. Available: https://www.imf.org/-/media/files/publications/imf-notes/2026/english/insea2026006.pdf

[4] Financial Action Task Force, "FATF recommendations: Recommendation 10 – customer due diligence," FATF/OECD, Tech. Rep., 2023. [Online]. Available: https://www.fatf-gafi.org/en/topics/fatf-recommendations.html

[5] European Parliament and Council of the European Union, "Regulation (EU) 2024/1183 of the european parliament and of the council establishing the European Digital Identity Framework (eIDAS 2.0)," Official Journal of the European Union, Tech. Rep., 2024. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/1183/oj/eng

[6] Financial Action Task Force, "Updated guidance for a risk-based approach to virtual assets and virtual asset service providers," Financial Action Task Force, Regulatory Guidance Framework, 2021, accessed: 2026-07-10. [Online]. Available: https://www.fatf-gafi.org/publications/fatfrecommendations/documents/guidance-rba-virtual-assets-2021.html

[7] European Parliament and Council of the European Union, "Regulation (EU) 2023/1114 on markets in crypto-assets (MiCA)," Official Journal of the European Union, Tech. Rep., 2023. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32023R1114

[8] S. Srinivasan, I. Karantaidou, F. Baldimtsi, and C. Papamanthou, "Batching, aggregation, and zero-knowledge proofs in bilinear accumulators," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. ACM, 2022, pp. 2719–2733.

[9] A. Ozdemir, R. S. Wahby, B. Whitehat, and D. Boneh, "Scaling verifiable computation using efficient set accumulators," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020, pp. 2075–2092. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/ozdemir

[10] W. G. Horner, "A new method of solving numerical equations of all orders, by continuous approximation," *Philosophical Transactions of the Royal Society of London*, vol. 109, pp. 308–335, 1819.

[11] Aztec Labs, "Noir: Domain-specific language for zero-knowledge proofs," Version 1.0.0-beta.15 (Commit 83245db), 2025. [Online]. Available: https://github.com/noir-lang/noir

[12] Aztec Protocol, "Barretenberg: UltraHonk proving system," Version 3.0.0-nightly.20251104, in aztec-packages monorepo, 2025. [Online]. Available: https://github.com/AztecProtocol/aztec-packages

[13] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC '85)*, 1985, pp. 291–304.

[14] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology — EUROCRYPT 2016, Part II*, ser. Lecture Notes in Computer Science, vol. 9666. Springer, 2016, pp. 305–326.

[15] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Paper 2019/953, 2019. [Online]. Available: https://eprint.iacr.org/2019/953

[16] C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac, "What makes Fiat-Shamir zkSNARKs (updatable SRS) simulation extractable?" in *Security and Cryptography for Networks (SCN)*. Springer, 2022, pp. 735–760.

[17] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Advances in Cryptology — EUROCRYPT '93*, ser. Lecture Notes in Computer Science, vol. 765. Springer, 1994, pp. 274–285.

[18] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology — CRYPTO 2002*, ser. Lecture Notes in Computer Science, vol. 2442. Springer, 2002, pp. 61–76.

[19] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Advances in Cryptology — EUROCRYPT '97*, ser. Lecture Notes in Computer Science, vol. 1233. Springer, 1997, pp. 480–494.

[20] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Topics in Cryptology — CT-RSA 2005*, ser. Lecture Notes in Computer Science, vol. 3376. Springer, 2005, pp. 275–292.

[21] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '87*, ser. Lecture Notes in Computer Science, vol. 293. Springer, 1988, pp. 369–378.

[22] Privacy and Scaling Explorations, "Semaphore: Zero-knowledge protocol for anonymous signaling," GitHub: semaphore-protocol/semaphore, 2019. [Online]. Available: https://semaphore.pse.dev

[23] R. Semenov, R. Storm, and A. Pertsev, "Tornado cash: Non-custodial private transactions on Ethereum," GitHub repository: tornadocash/tornado-core, 2019. [Online]. Available: https://github.com/tornadocash/tornado-core

[24] 0xbow, "Privacy pools: Compliant Ethereum privacy protocol," Mainnet launch, March 2025, 2025. [Online]. Available: https://www.0xbow.io

[25] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for Zero-Knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi

[26] U.S. Department of the Treasury, "Office of foreign assets control: Specially designated nationals and blocked persons list," Office of Foreign Assets Control, Regulatory Database Release, 2024, accessed: 2026-07-10. [Online]. Available: https://home.treasury.gov/policy-issues/financial-sanctions/specially-designated-nationals-and-blocked-persons-list-sdn-human-readable-lists

[27] United Nations Security Council, "The united nations security council consolidated sanctions list," United Nations Security Council Subsidiary Organs, Consolidated Manifest Release, 2025, accessed: 2026-07-10. [Online]. Available: https://www.un.org/securitycouncil/content/un-sc-consolidated-list

[28] H. Sarkhosh, U. Maroof, and D. Barradas, "The dark art of financial disguise in Web3: Money laundering schemes and countermeasures," *arXiv preprint arXiv:2509.21831*, September 2025. [Online]. Available: https://arxiv.org/abs/2509.21831

[29] L. Grassi, D. Khovratovich, and M. Schofnegger, "Poseidon2: A faster version of the Poseidon hash function," in *AFRICACRYPT 2023*, ser. Lecture Notes in Computer Science, vol. 14064. Springer, 2023, pp. 177–203.

[30] S. Trevisani, E. Andreeva, R. Bhattacharyya, and A. Roy, "Boosting efficiency and security in arithmetization-oriented hashing for zero-knowledge proof systems," Cryptology ePrint Archive, Paper 2026/1271, 2026. [Online]. Available: https://eprint.iacr.org/2026/1271

[31] J. Ernstberger, S. Chaliasos, G. Kadianakis, S. Steinhorst, P. Jovanovic, A. Gervais, B. Livshits, and M. Orrù, "zk-bench: A toolset for comparative evaluation and performance benchmarking of SNARKs," in *Security and Cryptography for Networks (SCN 2024)*. Springer-Verlag, 2024, p. 46–72.

[32] J. Liang, D. Hu, P. Wu, Y. Yang, Q. Shen, and Z. Wu, "SoK: Understanding zk-SNARKs: The gap between research and practice," in *34th USENIX Security Symposium*, 2025, arXiv:2502.02387.

[33] National Institute of Standards and Technology, "Privacy-Enhancing Cryptography (PEC) project," NIST Cryptographic Technology Group, Tech. Rep., 2025. [Online]. Available: https://csrc.nist.gov/projects/pec

[34] Andy, "PSE roadmap: 2025 and beyond," PSE blog, September 2025, posted 12 Sept. 2025. [Online]. Available: https://pse.dev/blog/pse-roadmap-2025

[35] R. Dahlberg, T. Pulls, and R. Peeters, "Efficient sparse Merkle trees: Caching strategies and secure (non-)membership proofs," in *Secure IT Systems — NordSec 2016*, ser. Lecture Notes in Computer Science, vol. 10014. Springer, 2016, pp. 199–215.

[36] D.-E. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," https://zips.z.cash/protocol/protocol.pdf, 2026, version 2026.7.0 [NU6.2].

[37] J. Kuszmaul, "Verkle trees," MIT PRIMES Program, mentored by Alin Tomescu, 2018. [Online]. Available: https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf
[38] V. Buterin, "Verkle trees," Personal blog, June 2021. [Online]. Available: https://vitalik.ca/general/2021/06/18/verkle.html
[39] M. Campanelli, M. Hall-Andersen, and S. H. Kamp, "Curve trees: Practical and transparent zero-knowledge accumulators," in *32nd USENIX Security Symposium*, 2023, pp. 4391–4408. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/campanelli
[40] M. Campanelli, M. Hall-Andersen, and S. H. Kamp, "Curve forests: Transparent zero-knowledge set membership with batching and strong security," in *Financial Cryptography and Data Security (FC 2025)*, ser. Lecture Notes in Computer Science, vol. 15751. Springer, 2025, pp. 219–229, IACR ePrint 2024/1647.
[41] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to IOPs and stateless blockchains," in *Advances in Cryptology — CRYPTO 2019*, ser. Lecture Notes in Computer Science, vol. 11692. Springer, 2019, pp. 561–586.
[42] M. Campanelli, D. Fiore, S. Han, J. Kim, D. Kolonelos, and H. Oh, "Succinct zero-knowledge batch proofs for set accumulators," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. ACM, 2022, pp. 455–469.
[43] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: Efficient, succinct, modular," *Designs, Codes and Cryptography*, vol. 91, no. 11, pp. 3457–3525, 2023, IACR ePrint 2019/1255.
[44] I. Damgård and N. Triandopoulos, "Supporting non-membership proofs with bilinear-map accumulators," Cryptology ePrint Archive, Paper 2008/538, 2008. [Online]. Available: https://eprint.iacr.org/2008/538
[45] S. Bayer and J. Groth, "Zero-knowledge argument for polynomial evaluation with application to blacklists," in *Advances in Cryptology — EUROCRYPT 2013*, ser. Lecture Notes in Computer Science, vol. 7881. Springer, 2013, pp. 646–663.
[46] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology – ASIACRYPT 2010*, ser. Lecture Notes in Computer Science, vol. 6477. Springer, 2010, pp. 177–194.
[47] J. Posen and A. A. Kattis, "Caulk+: Table-independent lookup arguments," Cryptology ePrint Archive, Paper 2022/957, 2022. [Online]. Available: https://eprint.iacr.org/2022/957
[48] H. Hafezi, G. Anthoine, M. Campanelli, and D. Fiore, "SoK: Lookup table arguments," Cryptology ePrint Archive, Paper 2025/1876, 2025. [Online]. Available: https://eprint.iacr.org/2025/1876
[49] C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems, "New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode," in *Advances in Cryptology — CRYPTO 2023*, ser. Lecture Notes in Computer Science, vol. 14083. Springer, 2023, pp. 507–539.
[50] W. J. Koh, "Gas and circuit constraint benchmarks of binary and quinary incremental Merkle trees using the Poseidon hash function," Ethereum Research Forum, 2020. [Online]. Available: https://ethresear.ch/t/gas-and-circuit-constraint-benchmarks-of-binary-and-quinary-incremental-merkle-trees-using-the-poseidon-hash-function/7446

APPENDIX A
SECURITY DEFINITIONS AND PROOFS

We refer to the polynomial accumulator scheme defined in §IV-F (Definition 4). Security properties are formalized with respect to that model.

Here we extend the condensed proofs of soundness and zero-knowledge from the main body (Theorems 1–3). We present zero-knowledge first, since the soundness proof's oracle hybrid relies on the simulator constructed there. Appendix A-D covers state binding, append-only consistency, public verifiability, and revocation support. Appendix A-E covers the privacy scope of self-sovereign proving, a non-default deployment variant.

Remark on Privacy Properties. Our security definitions emphasize membership authentication over stronger anonymity guarantees. The zero-knowledge property ensures that a proof reveals only membership validity beyond the public instance. We do not address the following.

- **Unlinkability.** The nullifier mechanism (§IV-F) provides unlinkability of the proof and nullifier across verifiers, under Poseidon's keyed indistinguishability (§III-D). It is not part of Theorems 1–3, and no bound in this appendix uses it. Its scope and what it does not cover, including the issuer's view and the other values a verifier observes, are stated in §IV-F.
- **Membership set hiding.** Our setting assumes public membership, consistent with privacy pools and compliance registries.
- **Forward security.** Both PEB constructions are append-only. Revocation requires epoch-level roster regeneration rather than in-place deletion.

A. Notation

Let $h = Poseidon(c)$ denote the public outer commitment to a batch coefficient vector $c = (c_0, \ldots, c_b)$ that defines the polynomial $P(x)$. Let $m \leq b$ denote the number of members in the batch and s_i a member's unhashed secret. Let $\rho(s)$ denote the root that a secret s contributes to the polynomial, so that $\rho(s) = s$ for PEB-A and $\rho(s) = Poseidon(s)$ for PEB-B, and write $\rho_i = \rho(s_i)$. Let s' denote the secret in the witness extracted by $\mathcal{E}$, as in Figure 1 and Definition 2. Let π denote a zk-SNARK proof over the public instance $x_{\text{pub}} = (h, \nu, k_V)$. Adversary $\mathcal{A}$ has adaptive access to $O_{\mathsf{GenProof}}(ref, k_V)$ with $ref \in \mathsf{Refs}$ and $k_V \in K_{\mathcal{A}}$ (Figure 1), issuing at most $q = \mathsf{poly}(\lambda)$ queries.

We use three error terms. The term $\varepsilon_{\mathsf{ZK}}$ bounds the advantage of any PPT distinguisher against the zero-knowledge property of the zk-SNARK on a single proof, so q oracle queries cost $q \cdot \varepsilon_{\mathsf{ZK}}$ [13], [15]. The term $\varepsilon_{\mathsf{SE}}$ bounds the probability that the simulation-extractor fails on an accepting fresh proof [16]. The term $\varepsilon_{\mathsf{Poseidon}}$ bounds the advantage of a PPT adversary against the collision resistance of Poseidon [29]. Game 3 uses it for both constructions and Game 4 uses it for PEB-B. We do not assume second-preimage resistance separately, because any second-preimage attack yields a collision, so collision resistance covers it (Games 3 and 4). We assume that UltraHonk is simulation-extractable (§II-B). Preimage resistance supports the witness-confidentiality arguments of §IV-H and is not used in the bounds below.

We write h^* for the commitment in the adversary's output $x^*_{\text{pub}} = (h^*, \nu^*, k^*_V)$. It only needs to be posted on $\mathcal{L}$ and does not have to be the latest commitment for its batch. The set S_{h^*} is the set of member secrets that the issuer committed under h^* at the time of posting, and c_{h^*} is the coefficient vector from which the honest issuer computed h^*. Freshness against the latest commitment is checked by the protocol (Step 7 of §IV-F) and is not part of the soundness experiment.

B. Extended Proof of Theorems 1–2 for Zero-Knowledge

a) Simulator Construction: By the zero-knowledge property of the underlying zk-SNARK [15], there exists a polynomial-time simulator $\mathsf{Sim}_{\mathsf{SNARK}}$ that takes only the public

statement (h, ν, k_V) and outputs a simulated proof π'. The full protocol simulator is defined as

$$\mathsf{Sim}(h, \nu, k_V) := \mathsf{Sim}_{\mathsf{SNARK}}(\mathcal{R}, (h, \nu, k_V))$$

where $\mathcal{R}$ is the circuit's constraint relation (Algorithm 2 for PEB-A and Algorithm 3 for PEB-B).

b) Game 0 (Real Proving Game): An honest prover generates $\pi \leftarrow \mathsf{GenProof}(pk, x_{\text{pub}}, x_{\text{priv}})$ using a valid private witness x_{priv}. A distinguisher $\mathcal{D}$ attempts to distinguish π from a simulated proof.

c) Game 1 (Simulated Game): The proof is generated without a witness via $\pi' \leftarrow \mathsf{Sim}(h, \nu, k_V)$.

d) Indistinguishability Reduction: Suppose $\mathcal{D}$ distinguishes π (Game 0) from π' (Game 1) with advantage δ. We construct a distinguisher $\mathcal{D}'$ against the zk-SNARK's zero-knowledge property. $\mathcal{D}'$ receives a challenge transcript for relation $\mathcal{R}$, either a real proof for the witness or a simulated proof from $\mathsf{Sim}_{\mathsf{SNARK}}$, and forwards it directly to $\mathcal{D}$. Both games give $\mathcal{D}$ the same public instance x_{pub}, including ν, so $\mathcal{D}'$ needs no further information. The transcript contains only π and no partial exposure of the witness, so $\mathcal{D}'$'s view is identical to $\mathcal{D}$'s, yielding advantage $\delta \leq \varepsilon_{\mathsf{ZK}}$. The protocol also sends *ref* and id_B to the verifier (Figure 2). These values are not produced by the SNARK prover, so they lie outside the guarantee of Definition 3 and are analyzed in §IV-F. Side-channel exposures or coefficient disclosures specific to the self-sovereign variant occur outside the SNARK transcript and are analyzed separately in Appendix A-E.

e) Conclusion: The argument is generic in $\mathcal{R}$ and applies identically to both constructions. The simulated transcript π' is computationally indistinguishable from a real transcript π except with probability at most $\varepsilon_{\mathsf{ZK}}$, establishing Theorems 1 and 2's zero-knowledge clauses.

C. Extended Proof of Theorems 1–2 for Soundness

We give the hybrid argument bounding $\mathcal{A}$'s overall forgery probability, using the simulator Sim constructed in Appendix A-B.

a) Game 0 (Real Adaptive Soundness Game): Game 0 is exactly $\mathsf{Exp}_{\Pi,\mathcal{A}}(1^\lambda)$ (Figure 1). After at most q adaptive oracle queries, $\mathcal{A}$ outputs $(x^*_{\text{pub}}, \pi^*)$ with $x^*_{\text{pub}} = (h^*, \nu^*, k^*_V)$. It wins if (a) h^* is posted on $\mathcal{L}$, (b) $\mathsf{Verify}(vk, x^*_{\text{pub}}, \pi^*) = 1$, (c) $(x^*_{\text{pub}}, \pi^*) \notin Q$, and (d) the extracted witness $s' \leftarrow \mathcal{E}(x^*_{\text{pub}}, \pi^*)$ has $s' \notin S_{h^*}$.

b) Game 1 (Oracle Hybrid Substitution): On each query (ref, k_V), the oracle still computes $x_{\mathsf{pub}}(ref, k_V) = (h, \nu, k_V)$ from the real secret and batch. We replace the proof with $\pi' \leftarrow \mathsf{Sim}(h, \nu, k_V)$, return $(x_{\mathsf{pub}}(ref, k_V), \pi')$, and add this pair to Q. We make this change one query at a time. Each substitution is indistinguishable except with probability $\varepsilon_{\mathsf{ZK}}$ by the zero-knowledge property established in Appendix A-B. Over q queries, the total distinguishing advantage is at most $q \cdot \varepsilon_{\mathsf{ZK}}$ by a standard hybrid argument over queries. After this step, $\mathcal{A}$'s view depends on the honest witness of a queried member only through the public instance.

c) Game 2 (Witness Extraction): By simulation-extractability of the zk-SNARK [16], an extractor $\mathcal{E}$ outputs a witness $w = (s', c, \mathsf{isKYCed})$ satisfying the circuit constraints whenever $\mathcal{A}$ produces an accepting fresh proof, except with probability at most $\varepsilon_{\mathsf{SE}}$. Because Game 1 has already made oracle answers simulator-generated, $\mathcal{E}$ can supply them without needing the honest witness itself.

d) Game 3 (Coefficient-Vector Collision): Conditioned on extraction, w contains a coefficient vector c satisfying $Poseidon(c) = h^*$. By the trust assumptions of §III-D, $\mathcal{L}$ authenticates the poster of each commitment, so only the issuer posts to $\mathcal{L}$. By condition (a), the issuer posted h^*, so $h^* = Poseidon(c_{h^*})$ for a vector c_{h^*} that the issuer holds. Let E_3 be the event $c \neq c_{h^*}$. A reduction $\mathcal{B}$ plays the issuer and builds every batch itself, so it knows c_h for every posted h and outputs the collision (c_{h^*}, c) whenever E_3 occurs. Hence $\Pr[E_3] \leq \varepsilon_{\mathsf{Poseidon}}$, with no loss across batches (see the Remark on the choice of assumption). If E_3 does not occur, then $c = c_{h^*}$ and $P(x)$ is the honest batch polynomial of h^*. Its roots are the member secrets for PEB-A and the hashed roots ρ_i for PEB-B.

e) Game 4 (Root Collision, PEB-*B Only):* Suppose E_3 does not occur. Then $P(Poseidon(s')) = 0$ implies $Poseidon(s') = \rho_k$ for some member s_k of the batch of h^*, because $P(x) = \prod_i (x - \rho_i)$ has exactly these roots. Let E_4 be the event $s' \neq s_k$ and $Poseidon(s') = Poseidon(s_k)$. A reduction $\mathcal{B}'$ plays the issuer and generates every member secret itself, so it knows s_k and outputs the collision (s', s_k) whenever E_4 occurs. Hence $\Pr[E_4] \leq \varepsilon_{\mathsf{Poseidon}}$, with no loss across members or batches (see the Remark on the choice of assumption). This game is omitted for PEB-A, where roots are unhashed secrets.

f) Remaining Case: Suppose extraction succeeds and neither E_3 nor (for PEB-B) E_4 occurs. For PEB-A, the roots of $P(x)$ are exactly the member secrets, so $P(s') = 0$ gives $s' \in S_{h^*}$. For PEB-B, the absence of E_4 gives $s' = s_k$ for some k, so $s' \in S_{h^*}$. Both contradict condition (d), so $\mathcal{A}$ cannot win in this case. The standard $m/|\mathbb{F}|$ polynomial root-bound is inapplicable here, as $P(x)$ is pinned to a fixed vector rather than evaluated at a fresh random verifier challenge point.

g) Conclusion: If $\mathcal{A}$ wins, then extraction fails, or E_3 occurs, or (for PEB-B) E_4 occurs. Union-bounding over these events, together with the $q \cdot \varepsilon_{\mathsf{ZK}}$ loss of Game 1, bounds $\mathcal{A}$'s winning probability by at most $q \cdot \varepsilon_{\mathsf{ZK}} + \varepsilon_{\mathsf{SE}} + \varepsilon_{\mathsf{Poseidon}}$ for PEB-A, and by at most $q \cdot \varepsilon_{\mathsf{ZK}} + \varepsilon_{\mathsf{SE}} + 2\varepsilon_{\mathsf{Poseidon}}$ for PEB-B. The factor 2 combines Game 3's coefficient-vector collision with Game 4's root collision. Both constructions use the zero-knowledge and simulation-extractability of the SNARK and the collision resistance of Poseidon.

h) Remark (Choice of Assumption): Games 3 and 4 bound two events that are, in practice, preimage or second-preimage attacks on Poseidon. We bound them by collision resistance because the reduction plays the issuer and therefore holds both inputs of the resulting pair. This is a stronger assumption than second-preimage or preimage resistance, so the proven bound is conservative relative to the practical attacks. We chose it

for three reasons. First, it is the assumption on which the soundness of the IMT and SMT baselines rests [21], [35], so PEB requires no assumption beyond them. Second, a proof of second-preimage or preimage resistance requires a stronger experiment. It needs a corruption oracle that fixes which batches $\mathcal{A}$ knows, a guess over the posted batches, and an argument that the structured coefficient vectors satisfy a definition stated for uniform inputs. Third, the stronger assumption costs no practical security. Generic collision search on the output of Poseidon remains computationally infeasible at the parameters we use. We leave a tighter analysis under weaker assumptions to future work.

D. Other Security Properties

The definitions below are stated for the generic syntax of §IV-B, so they apply to PEB-A, PEB-B, and the IMT/SMT baselines. In each case x_{pub} contains the commitment named in the definition.

Definition 5 (State Binding)**.** *A membership accumulator satisfies **state binding** if for any PPT adversary that outputs two distinct member sets $S \neq S'$, the commitments $C \leftarrow \mathsf{Commit}(S)$ and $C' \leftarrow \mathsf{Commit}(S')$ satisfy $\Pr[C = C'] \leq \mathsf{negl}(\lambda)$.*

Theorem 4 (State Binding of PEB)**.** *Under collision resistance of Poseidon [29], both* PEB-*A and* PEB-*B satisfy Definition 5.*

Proof. Fix $S \neq S'$ with respective monic characteristic polynomials P, P' of degree $\leq b$ (§IV-D). For PEB-A, the roots are the secrets. For PEB-B, the roots are the hashed secrets. In the second case, if the root multisets coincide while $S \neq S'$, then two distinct secrets share a hash, which is a collision for Poseidon. Otherwise the root multisets differ. Since a monic polynomial over a field is uniquely determined by its root multiset, $P \neq P'$, so the padded coefficient vectors $c = (c_0, \ldots, c_b)$ and $c' = (c'_0, \ldots, c'_b)$ differ in at least one position. If $C = Poseidon(c)$ and $C' = Poseidon(c')$ collide despite $c \neq c'$, this is by definition a collision for Poseidon, occurring with probability at most $\mathsf{negl}(\lambda)$ under collision resistance. □

Definition 6 (Append-Only Consistency)**.** *A membership accumulator satisfies **append-only consistency** if it meets the following condition up to structure capacity. Let $\mathbf{c}$ be a coefficient vector obtained by applying* Add *once per element of a set S from $\mathbf{c}^{\mathsf{init}}$, with $C \leftarrow \mathsf{Commit}(\mathbf{c})$, and let $r \notin S$. Given $\mathbf{c}' \leftarrow \mathsf{Add}(\mathbf{c}, r)$ and $C' \leftarrow \mathsf{Commit}(\mathbf{c}')$, for every $s \in S$ there exist a public instance x'_{pub} containing C' and a witness x'_{priv} such that $\mathsf{Verify}(vk, x'_{pub}, \pi') = 1$ for $\pi' \leftarrow \mathsf{GenProof}(pk, x'_{pub}, x'_{priv})$.*

Theorem 5 (Append-Only Consistency of PEB)**.** *Both* PEB-*A and* PEB-*B satisfy Definition 6, up to batch capacity b.*

Proof. Let $\mathbf{c}$ result from applying Add once per element of S from $\mathbf{c}^{\mathsf{init}}$, with $C \leftarrow \mathsf{Commit}(\mathbf{c})$. Fix $r \notin S$, $\mathbf{c}' \leftarrow \mathsf{Add}(\mathbf{c}, r)$, and $C' \leftarrow \mathsf{Commit}(\mathbf{c}')$, computing $P'(X) = P(X)(X - \rho(r))$. For any $s \in S$, $P'(\rho(s)) = P(\rho(s)) \cdot (\rho(s) - \rho(r)) = 0$, so the root of s remains a root of P'. This is $P'(s) = 0$ for PEB-A and $P'(Poseidon(s)) = 0$ for PEB-B. The prover generates $\pi' \leftarrow \mathsf{GenProof}(pk, (C', \nu, k_{\mathcal{V}}), (s, \mathbf{c}', \text{isKYCed}))$. Horner evaluation of P' at $\rho(s)$ returns 0, so $\mathsf{Verify}(vk, (C', \nu, k_{\mathcal{V}}), \pi') = 1$ by the same argument as Theorem 1 (PEB-A) or Theorem 2 (PEB-B). This holds while $\deg(P)+1 \leq b$. Capacity is reached at $\deg(P) = b$. □

Definition 7 (Public Verifiability)**.** *An accumulator is **publicly verifiable** if* Verify *is a deterministic function of only the verification key vk, the public instance x_{pub} (which contains the commitment C), and the proof π, with no secret key, witness, or trapdoor as input.*

Theorem 6 (Public Verifiability of PEB)**.** *Both* PEB-*A (Algorithm 2) and* PEB-*B (Algorithm 3) satisfy Definition 7.*

Proof. The verifier of PEB takes only vk, $x_{\text{pub}} = (h, \nu, k_{\mathcal{V}})$, and π for verification, requiring no secret witness or trapdoor (Definition 4). The SNARK verifier handles the process using these public values and π. The freshness check of Step 7 in §IV-F reads only the public log L. □

Definition 8 (Revocation Support)**.** *An accumulator supports **revocation** if there is an efficient procedure to produce $C'' \leftarrow \mathsf{Commit}(S \setminus \{r\})$ for a revoked member $r \in S$, such that stale proofs for C are rejected once C'' supersedes C, while every unaffected member $s \in S \setminus \{r\}$ can generate a fresh proof that verifies under a public instance containing C''.*

Theorem 7 (Revocation Support of PEB)**.** *Both* PEB-*A and* PEB-*B satisfy Definition 8 via epoch-boundary regeneration, with stale proofs against a revoked member rejected with overwhelming probability.*

Proof. To produce C'', we rebuild the coefficient vector $\mathbf{c}''$ from $\mathbf{c}^{\mathsf{init}}$ by applying Add for each element in $S \setminus \{r\}$, followed by setting $C'' \leftarrow \mathsf{Commit}(\mathbf{c}'')$. No algorithms beyond Add and Commit are necessary. By Theorem 5, the root of each $s \in S \setminus \{r\}$ remains a root of the polynomial P'' during this process. Since $S \setminus \{r\} \neq S$, Theorem 4 ensures $C'' \neq C$ with probability at most $\mathsf{negl}(\lambda)$. Thus, the event-log check (§IV-F) rejects any stale proof tied to C except with probability at most $\mathsf{negl}(\lambda)$. For each $s \in S \setminus \{r\}$, we can compute a valid proof under C'' since the root of s remains a root of P'', as shown by Theorems 1 and 2. Note that PEB does not allow in-place deletion and requires complete epoch regeneration in line with the noted forward-security scope. Because C'' differs from C, every proof against C is rejected, including the proofs of unaffected members. Revoking one member therefore requires all remaining members of the batch to obtain fresh proofs. This shows revocation is possible in principle. We do not evaluate it here (§VIII-A). □

Applicability to Traditional Merkle Trees. State binding, append-only consistency, public verifiability, and revocation support (Definitions 5–8) are standard, well-established properties of Merkle-tree accumulators under a collision-resistant hash function [21], [35] and transfer directly to our IMT/SMT instantiation. The nullifier/verifier_key wrapper (§IV-F) is layered

outside the root computation and exposes no additional public input beyond $rt, \nu, k_{\mathcal{V}}$, so it does not alter the underlying tree's binding, consistency, verifiability, or revocation guarantees.

E. Self-Sovereign Proving: A Non-Default Threat Model

Section IV-F outlines a holder-distributed variant of PEB-B. In this version, the issuer shares the coefficient vector with all batch holders during registration and again at closure. After this point, each holder can independently execute Algorithm 3 without any further involvement from the issuer, provided they conduct the same event log check. We analyze this variant's security here. This is not our recommended deployment.

a) Threat model change: This relaxes the issuer-as-prover custody assumption of §III-D for PEB-B. Instead of a single honest prover, up to b mutually distrusting holders each legitimately possess the full coefficient vector $\{c_0, \ldots, c_b\}$ for their batch. From the view of any one holder, the other members number at most $b - 1$.

b) Residual exposure: Correctness and soundness (Definitions 1, 2) remain unaffected, since Algorithm 3 used by PEB-B is unchanged and its guarantees do not depend on who invokes it. This variant introduces three residual exposures relative to the issuer-as-prover default.

- **Coefficient-hash disclosure.** Each holder now learns the hashed roots $\{\rho_j\}_{j \neq i}$ of the other members in their batch, which is a strictly weaker guarantee than the default issuer-as-prover model, where no such information is shared. Recovering any s_j from $\rho_j = Poseidon(s_j)$ requires inverting Poseidon or searching the secret space, so this variant is secure only if the secrets have high entropy. The PEB-A variant is not compatible with this approach at all (§IV-H).
- **Side-channel exposure.** Distributing proof generation across up to b holder-owned devices with unknown security postures increases exposure to local side-channel attacks, such as power analysis or cache timing, against the private witness during proof generation. The issuer-as-prover default confines this risk to a single, hardenable environment. This exposure is a property of where proving occurs, not of the proof transcript, so it is orthogonal to the transcript-level zero-knowledge guarantee of Definition 3 (Appendix A-B).
- **Higher revocation cost.** Revocation is more costly under this variant, since the updated coefficient vector must be redistributed to all remaining holders, at most $b - 1$ of them, rather than only the on-chain commitment changing.

Addressing this gap is left as an open question (§VIII-A).

APPENDIX B
PERFORMANCE BREAKDOWN

Table V decomposes initialization cost into constituent operations, supplementing the asymptotic analysis of Table I. Table VI reports measured population, proof-generation, and verification time at three representative population sizes across all four IMT/SMT × PEB-A/PEB-B pairings, supplementing the aggregate speedup figures of Section V. Values are not comparable across pairing blocks, as each is an independently benchmarked session.

TABLE V: Operation breakdown at $n = 2^{20}$.

Step	IMT (incr.)	SMT (bulk)	PEB-A	PEB-B
Per-insertion hashes	21 Poseidon calls	—	—	—
Root hashes ($\rho_i = Poseidon(s_i)$)	—	—	—	2^{20}
Total hashes at $n = 2^{20}$	22,020,096	2,097,151	8,192	1,056,768
Field ops (coeff. computation, upper bound)	—	—	$8{,}192 \times 128^2 = 134{,}217{,}728$	
Mapping insertions	—	—	2^{20}	

The dominant PEB-A cost is the calculation of the polynomial coefficients. Each batch commitment hash absorbs 129 field elements, so one counted commitment hash costs more than a two-input Merkle node hash. The PEB-A hash count is therefore a lower bound on its hashing work, and it remains small. PEB-B additionally computes one Poseidon evaluation per member to obtain $\rho_i = Poseidon(s_i)$ before root insertion (§IV-E). This adds 2^{20} hashes. These are far fewer operations than the field operations, but each hash costs far more. In the IMT vs. PEB-A block of Table VI, population takes 3,128,231.70 ms for 22,020,096 hashes, which is about 0.142 ms per hash. At this rate, 2^{20} root hashes take about 149 s. The PEB-A population in the same block takes about 25 s in total. Root hashing therefore dominates PEB-B population time. This matches the PEB-B values in Table VI, such as 175,917.35 ms at $n = 2^{20}$, although those values come from a separate session. The total-hash count of SMT is a direct consequence of single-pass bottom-up construction (§IV-C) and requires no path recomputation.

A. Structural Origin of the Cost Asymmetry

Population cost is measured off-circuit. The speedup therefore reflects the native cost of a Poseidon evaluation versus a field multiplication in our implementation. It is not an in-circuit constraint effect. In-circuit costs appear in proof generation, where PEB is slower (Table VI).

Any cryptographic hash suitable for a Merkle accumulator, such as Poseidon [25], Poseidon2 [29], and Anemoi [49], requires a multi-round permutation to resist algebraic and interpolation attacks [25]. Recent efficiency optimizations reduce constant factors (up to 2× in software, ∼60% in proving) but do not eliminate round complexity [30]. Consequently, every hash evaluation costs many field multiplications. Conversely, PEB enforces root membership via the SNARK constraint system rather than cryptographic mixing. Raw field multiply-accumulate operations are its cost floor.

Substituting hardware-accelerated, non-algebraic hashes (e.g., SHA-256, Blake3) to accelerate native population shifts the bottleneck. IMT and SMT must re-evaluate these paths in-circuit, and bit-oriented operations inflate arithmetization constraints and degrade proof generation [50], [25].

In hash counts, the gap exceeds an order of magnitude for PEB-A against both baselines and for PEB-B against IMT (Table V). The PEB-B against SMT pairing is the exception. Both sides pay mainly for hashing, with 2,097,151 hashes for SMT and 1,056,768 for PEB-B, a ratio of about 2.0. The measured speedup of about 1.8× (§V) is close to this ratio. We

TABLE VI: Paired metric comparison at representative population sizes, all four pairings, Run 1. Each block is a single independent benchmarking run; values are not comparable across blocks.

(a) IMT comparisons

Pairing	Metric	n	Baseline	PEB
IMT vs. PEB-A (incremental)	Pop. total (ms)	1,024	3497.98	22.93
		65,536	195146.87	1250.42
		1,048,576	3128231.70	24761.92
	Proof gen. (ms)	1,024	95.05	128.84
		65,536	100.62	132.75
		1,048,576	105.32	136.16
	Verif. (ms)	1,024	13.21	15.82
		65,536	17.13	17.52
		1,048,576	20.08	22.77
IMT vs. PEB-B (incremental)	Pop. total (ms)	1,024	3230.11	178.85
		65,536	200345.27	10719.60
		1,048,576	3209287.90	175917.35
	Proof gen. (ms)	1,024	98.08	132.13
		65,536	94.71	121.33
		1,048,576	104.54	138.64
	Verif. (ms)	1,024	13.26	13.65
		65,536	14.32	14.80
		1,048,576	21.46	21.58

(b) SMT comparisons

Pairing	Metric	n	Baseline	PEB
SMT vs. PEB-A (bulk)	Pop. total (ms)	1,024	314.61	21.49
		65,536	18457.89	1175.44
		1,048,576	296417.28	18592.26
	Proof gen. (ms)	1,024	97.22	128.11
		65,536	97.55	132.34
		1,048,576	105.58	139.49
	Verif. (ms)	1,024	12.80	14.00
		65,536	14.86	14.90
		1,048,576	21.57	22.49
SMT vs. PEB-B (bulk)	Pop. total (ms)	1,024	303.60	168.25
		65,536	18552.19	10421.71
		1,048,576	296637.08	166714.17
	Proof gen. (ms)	1,024	96.52	123.82
		65,536	96.32	127.65
		1,048,576	96.89	128.08
	Verif. (ms)	1,024	12.66	13.72
		65,536	15.10	15.77
		1,048,576	20.86	20.47

highlight this sensitivity explicitly rather than claim an absolute lower bound for that pairing. This suggests the advantage holds for any hash that costs much more than a field operation, but our measurements cover one implementation. They do not establish dominance over proof-generation overhead.