A Merkleized Batch Attestation Scheme with Selective Disclosure and Compliance-by-Design

Harry Willson Potter — Independent Research

Abstract

We present a formal treatment of Merkleized batch attestation, a cryptographic protocol that enables issuers to commit a set of credentials to a single on-chain root while allowing subjects to disclose individual attestations selectively. The scheme resolves what we term the *efficiency-privacy-compliance trilemma* in decentralized credential systems: prior constructions sacrifice at least one of on-chain efficiency (storing each attestation independently), privacy (revealing the full attestation set upon verification), or regulatory compliance (lacking hooks for identity screening and data minimization mandates). We define a unified syntax for the scheme as a tuple of algorithms (Issue, Aggregate, Store, Verify) and prove three security properties: batch integrity (root authenticity under adaptive chosen-message attacks), selective disclosure soundness (no adversary can convincingly reveal a leaf not in the committed set), and beacon unlinkability (on-chain identifiers reveal no linkage between a subject's multiple attestations). We then analyze three batch strategies—Periodic, On-demand, and Hybrid—and prove optimality bounds for each. A formal design pattern, Compliance-Guard, is introduced to compose attestation logic with regulatory predicates (KYC, sanctions screening, data minimization) while preserving the scheme's security guarantees. Numerical analysis estimates gas costs across batch sizes and compares verification overhead against zero-knowledge-based alternatives. The results demonstrate that Merkleized batching achieves sub-linear on-chain cost, constant verification latency for compliant verifiers, and information-theoretic selective disclosure without recursive proof composition.

---

1 Introduction

Decentralized credential systems promise to restore user control over personal attestations—employment records, educational certificates, professional licenses—by replacing siloed registries with blockchain-anchored commitments. A growing body of systems [7,

12, 20, 27] implements this vision, yet each confronts a persistent tension among three design goals.

Efficiency. On-chain storage is expensive. Ethereum gas costs, for example, make per-attestation `SSTORE` operations prohibitive at scale. A university issuing 10,000 diplomas or a human resources department attesting 50,000 employment records faces costs that dwarf the value of the credentials themselves. Prior work mitigates this through batching [27]—issuing n credentials under a single Merkle root—but does so in a way that either sacrifices selective disclosure or requires heavy cryptographic machinery (zero-knowledge proofs) that shifts cost rather than eliminating it.

Privacy. Selective disclosure—the ability of a subject to reveal a specific attestation without exposing the remainder of the set—is a first-order requirement under privacy regulations such as the GDPR [9] and the PIPL [22]. A candidate verifying a single prior employment date should not be forced to disclose every past position, salary figure, or termination reason. Existing batch-issuance schemes, however, typically release the full credential set to the verifier, as the on-chain commitment binds all leaves simultaneously and the system provides no mechanism for partial revelation.

Compliance. Regulatory frameworks increasingly mandate that credential systems gate participation behind identity verification (KYC), perform sanctions screening, and enforce data minimization by design [3, 5, 22]. A scheme that achieves efficiency and privacy but cannot accommodate these compliance hooks is unusable in regulated environments—precisely the environments where credential fraud and privacy violations are most consequential.

We call this the *efficiency-privacy-compliance trilemma*: no existing system satisfies all three desiderata simultaneously. Blockcerts [27] achieves efficient batch issuance but lacks selective disclosure and compliance hooks. Verifiable Credentials with zero-knowledge proofs [29] enable selective disclosure at the cost of on-chain verification overhead and complex trusted setup. Permissioned frameworks such as FISCO BCOS [13] embed compliance at the ledger level but require all participants to operate within a single trust domain, undermining the decentralization that motivates the use of blockchain in the first place.

1.1 Contributions

This paper provides a formal foundation for a Merkleized batch attestation scheme that resolves the trilemma. Our contributions are as follows.

1. **Formal syntax and security definitions.** We define the Merkleized batch attestation scheme as a tuple $(\mathsf{Setup}, \mathsf{Issue}, \mathsf{Aggregate}, \mathsf{Store}, \mathsf{Verify})$ and specify three security games: batch integrity (Game G_{BI}), selective disclosure soundness (Game G_{SD}), and beacon unlinkability (Game G_{UL}). We prove that the scheme satisfies all three properties under standard assumptions (collision resistance of SHA-256 and the random oracle model).
2. **Optimal batch strategies.** We analyze three batch strategies—Periodic, On-demand, and Hybrid—and derive bounds on their worst-case latency, storage overhead, and verification cost. We prove that the Hybrid strategy is Pareto-optimal with respect to the latency–cost trade-off.
3. **ComplianceGuard formal pattern.** We introduce a formal design pattern that composes attestation logic with compliance predicates. The pattern guarantees that no attestation state transition occurs unless the invoking party satisfies issuer-defined compliance rules (e.g., KYC completion, sanctions list clearance), and that the predicate evaluation itself is auditable via on-chain events.
4. **Comparative analysis.** We provide numerical estimates of on-chain gas consumption for batch sizes from 1 to 1024, identify the crossover point at which Merkleized batching outperforms per-attestation storage, and compare verification overhead against a representative zero-knowledge-based scheme (zk-SNARKs over the same credential relation).

1.2 Roadmap

Section 2 surveys related work in blockchain credentials, Merkle-based privacy, and compliance architectures. Section 3 presents the system model, including roles, ledger abstraction, and the adversary model. Section 4 defines the protocol syntax formally. Section 5 gives the security definitions and proofs. Section 6 analyzes the three batch strategies. Section 7 formalizes the ComplianceGuard design pattern. Section 8 provides numerical analysis. Section 9 discusses limitations and future work. Section 10 concludes.

2 Related Work

2.1 Blockchain-Anchored Credentials

The earliest and most influential deployment of Merkleized credentials on a public blockchain is Blockcerts [27], developed at the MIT Media Lab. Blockcerts batches multiple certificates under a single

Merkle root recorded on the Bitcoin or Ethereum blockchain. A recipient receives a JSON document containing both the credential and a Merkle proof; verification requires checking the proof against the on-chain root. Blockcerts achieves $O(1)$ on-chain cost per batch and $O(\log n)$ verification, but it does not support selective disclosure. The entire credential payload (including all fields of all certificates in the batch) is hashed and revealed as a unit. Furthermore, Blockcerts provides no built-in compliance layer; KYC, sanctions screening, and data minimization must be implemented ad hoc by each deploying institution.

Verifiable Credentials (VCs) [29] define a W3C-standard data model for expressing attestations signed by an issuer. The VC model supports selective disclosure through zero-knowledge proofs (ZKP-VCs): a subject can present a proof that a specific claim (e.g., "employment date $\leq$ 2020-01-01") satisfies the issuer's signature without revealing the claim value itself. However, ZKP-VCs impose significant verification costs on-chain (or require off-chain verification, which defeats the purpose of blockchain-anchored trust), and they typically rely on a trusted setup or a structured reference string. Moreover, the VC ecosystem assumes a Verifiable Data Registry (VDR) for resolving issuer DIDs and revocation status; there is no standard for making the VDR itself efficient or compliant.

FISCO BCOS [13] is a permissioned blockchain framework developed by WeBank that supports group signatures, ring signatures, and homomorphic encryption through its WeIdentity DID module. Because FISCO BCOS is permissioned, compliance (KYC, AML) can be enforced at the node admission level. However, this comes at the cost of decentralization: the consortium of nodes that govern the ledger can collude to alter attestation records, and subjects have no guarantee that their credentials survive a change in consortium membership.

Other systems include uPort [16] (self-sovereign identity on Ethereum, now deprecated), Sovrin [14] (a purpose-built permissioned ledger for identity), and the recent EBSI (European Blockchain Services Infrastructure) [8], which defines a pan-European framework for verifiable credentials. None of these systems simultaneously provide batch efficiency, selective disclosure, and a formal compliance composition layer, confirming the presence of the trilemma.

2.2 Merkle Trees for Privacy

Merkle trees [18] have been used as a privacy primitive in numerous blockchain contexts beyond credentials. In healthcare, MedRec [1]

uses Merkle trees to commit patient records while enabling patients to grant granular access to providers. In supply chain, Tian [25] proposes a Merkle-based traceability system that reveals only the relevant segment of the supply chain to each auditor. In blockchain scaling, Merkle proofs are the foundation of payment verification in Bitcoin [19] and of stateless clients in Ethereum [24].

The common pattern in these works is the use of a Merkle tree to commit a set S of size n such that a prover can later convince a verifier of the statement "$x \in S$" using a proof of size $O(\log n)$. Our work adopts the same pattern but applies it to the specific structural constraints of career attestations: each subject has a sequence of time-ordered attestations from multiple independent issuers, and the verifier may need only a single attestation from the sequence. We formalize this setting and prove security properties that are specialized to the attestation context.

2.3 Compliance Patterns in Decentralized Systems

Compliance in decentralized credential systems has been addressed primarily through architectural patterns rather than formal methods. The "layered architecture" approach [4] separates credential issuance from compliance verification by introducing a compliance oracle or gateway that attests to a subject's compliance status. Holy and Beer [11] propose a "Compliance Manager" smart contract that gates access to credential operations based on an external identity oracle. In regulated DeFi, the Travel Rule Protocol (TRP) [26] defines a message-layer standard for compliance information exchange.

Our ComplianceGuard pattern formalizes this family of approaches. Unlike prior work, which treats compliance as an external module orthogonal to the credential protocol, we define compliance as a *predicate transformer* on the attestation scheme itself: the ComplianceGuard wraps the Issue and Verify algorithms and ensures that every state transition is auditable and predicate-satisfying. This yields a composable design in which security properties of the underlying scheme are preserved under the compliance wrapper.

2.4 Zero-Knowledge Alternatives

A natural alternative to Merkle-based selective disclosure is to use zero-knowledge proofs [10]. In this approach, the issuer signs a set of attestations, and the subject proves in zero knowledge that a particular attestation is in the signed set, without revealing the set. Schemes such as zk-creds [17] and anonymous credentials [6] follow this paradigm.

The trade-off is primarily computational. A zk-SNARK for the set membership relation requires a trusted setup (or a transparent setup with larger proofs, e.g., STARKs [2]), and on-chain verification of a SNARK costs on the order of 200,000–500,000 gas in current EVM implementations [21]. A Merkle proof, by contrast, costs approximately $c \cdot \log n$ gas, where c is the cost of a SHA-256 hash computation (approximately 3,000–5,000 gas in Solidity). For $n \leq 1024$, this is under 50,000 gas. The Merkle approach therefore has a significant cost advantage for small to medium batch sizes, which covers the vast majority of attestation scenarios (e.g., an individual's career history of 5–15 attestations, a university course cohort of 100–500 students). We quantify this comparison in Section 8.

3 System Model

3.1 Roles and Ledger Abstraction

The system comprises four roles.

Issuer ($\mathcal{I}$). An entity that creates attestations about a Subject. Examples: an employer attesting to a former employee's dates of service; a university attesting to a graduate's degree. Each Issuer has a key pair $(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}})$.

Subject ($\mathcal{S}$). The entity to whom attestations pertain. A Subject controls one or more *beacons*—pseudonymous on-chain identifiers—through which attestations are bound. We model a beacon as $\mathsf{beacon} = H(\mathsf{phone} \parallel \mathsf{salt})$, where H is a cryptographic hash function and salt is a subject-chosen blinding factor.

Verifier ($\mathcal{V}$). An entity that checks the validity of an attestation claimed by a Subject. Example: a prospective employer verifying a candidate's prior employment. A Verifier holds the on-chain state (or a trusted copy thereof) but does not hold any Issuer secret key.

Compliance Authority ($\mathcal{C}$). A trusted (or distributed) entity that evaluates compliance predicates. Typical predicates include: "Subject $\mathcal{S}$ has completed KYC," "Subject $\mathcal{S}$ is not on a sanctions list," "Issuer $\mathcal{I}$ is a registered attestor." The Compliance Authority publishes signed predicate results that are consumed by the compliance wrapper.

Ledger abstraction. We assume a public, append-only, Byzantine fault-tolerant ledger $\mathcal{L}$ that supports: - $\mathcal{L}.\mathsf{put}(\mathsf{key}, \mathsf{value})$: persistent key-value storage, costing proportional to the byte-length of value. - $\mathcal{L}.\mathsf{get}(\mathsf{key})$: retrieval of the latest value for a key. - $\mathcal{L}.\mathsf{emit}(\mathsf{event})$: emission of an auditable event log.

The ledger corresponds to a smart-contract platform such as Ethereum. Our analysis assumes EVM gas accounting, but the scheme is platform-agnostic.

3.2 Adversary Model

We consider a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ with the following capabilities:

- **Network-level.** $\mathcal{A}$ can observe all transactions submitted to $\mathcal{L}$, including attestation storage operations, beacon registrations, and verification requests. $\mathcal{A}$ can reorder, delay, or censor transactions, but cannot prevent eventual inclusion if the transaction is valid (liveness). $\mathcal{A}$ cannot break the underlying cryptographic primitives (collision resistance of SHA-256, existential unforgeability of the signature scheme).
- **Key compromise.** $\mathcal{A}$ may compromise the secret keys of a subset of Issuers and Subjects. Compromised keys are available to $\mathcal{A}$ for signing. Honest parties' keys remain secret.
- **Compliance bypass.** $\mathcal{A}$ may attempt to invoke attestation operations without satisfying compliance predicates. The ComplianceGuard must prevent this.
- **Goal.** $\mathcal{A}$'s goal is to violate one of the three security properties: batch integrity (forge an attestation under an honest Issuer's root), selective disclosure soundness (convince a Verifier of a false attestation), or beacon unlinkability (link two attestations to the same Subject).

We assume the ledger's consensus mechanism provides common prefix and chain quality properties [23], ensuring that any transaction visible to an honest party is eventually visible to all honest parties and that $\mathcal{A}$ cannot retroactively alter the ledger history.

4 Protocol Syntax

Let λ be the security parameter. Let $H : \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a cryptographic hash function (modeled as a random oracle in security proofs). Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be an existentially unforgeable signature scheme.

A Merkleized batch attestation scheme Π is a tuple of five PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. On input security parameter λ, output public parameters pp (e.g., the choice of H, the tree arity, the batch strategy configuration).

- $\mathsf{Issue}(\mathsf{sk}_{\mathcal{J}}, \mathsf{beacon}, m) \rightarrow (\mathsf{att}, \sigma)$. On input an Issuer secret key $\mathsf{sk}_{\mathcal{J}}$, a Subject beacon beacon, and an attestation payload $m \in \{0,1\}^*$, output an attestation $\mathsf{att} = (\mathsf{beacon}, m)$ and a signature $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_{\mathcal{J}}, \mathsf{att})$.
- $\mathsf{Aggregate}(\mathcal{T}) \rightarrow (r, \Pi)$. On input a set $\mathcal{T} = \{\mathsf{att}_1, \ldots, \mathsf{att}_n\}$ of attestations, where each $\mathsf{att}_i = (\mathsf{beacon}_i, m_i, \sigma_i)$, verify each signature σ_i under the respective Issuer's public key. If all signatures verify, construct a Merkle tree over the leaf values $\ell_i = H(\mathsf{att}_i)$:

  $$r \leftarrow \mathsf{MerkleRoot}(\ell_1, \ldots, \ell_n), \qquad \Pi \leftarrow \{\mathsf{MerklePath}(\ell_i) \mid i \in [n]\}.$$

  Output the root r and the set of Merkle proofs Π.
- $\mathsf{Store}(r, \mathsf{beacon}_*)$. On input a root r and (optionally) a beacon commitment beacon_*, write r to the ledger at the location indexed by beacon_*. Output a ledger confirmation txHash.
- $\mathsf{Verify}(\mathsf{beacon}, m, \sigma, \pi, \mathsf{pk}_{\mathcal{J}}) \rightarrow \{0,1\}$. On input a beacon beacon, an attestation payload m, a signature σ, a Merkle proof π, and an Issuer public key $\mathsf{pk}_{\mathcal{J}}$, proceed as follows:
  1. Check $\Sigma.\mathsf{Verify}(\mathsf{pk}_{\mathcal{J}}, (\mathsf{beacon}, m), \sigma) = 1$.
  2. Compute $\ell = H(\mathsf{beacon}, m, \sigma)$.
  3. Retrieve the on-chain root r at the beacon's index.
  4. Check $\mathsf{MerkleVerify}(r, \ell, \pi) = 1$. Output 1 if all checks pass, else 0.

Remark 1 (Beacon indexing). The root r may be stored at a ledger key derived from beacon, or at a key derived from the subject's on-chain account address. The former binding (beacon-root) provides unlinkability as formalized in Section 5.3; the latter (address-root) is simpler but linkable. Our security analysis assumes the beacon-indexed variant.

Remark 2 (Batch granularity). $\mathsf{Aggregate}$ may operate over attestations issued to multiple subjects or a single subject. In the multi-subject case, each attestation is a leaf, and each subject holds only the subset of Merkle proofs corresponding to their own attestations. In the single-subject case, all leaves belong to the same subject, and the root serves as a compact summary of that subject's attestation history. The same Verify algorithm suffices for both cases.

5 Security Definitions and Proofs

We define three security games. In each, the adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ that simulates the honest parties. Let Π be a Merkleized batch attestation scheme as defined in Section 4.

5.1 Batch Integrity

Batch integrity captures the requirement that no adversary can produce a valid attestation under an honest Issuer's root unless the Issuer indeed signed that attestation.

Game G_{BI}**.** The challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda)$ and generates a key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ for the target Issuer. $\mathcal{A}$ is given pk^* and oracle access to $\mathcal{O}_{\mathsf{Issue}}(\cdot,\cdot)$, which on input (beacon_i, m_i) returns $(\mathsf{att}_i, \sigma_i) \leftarrow \mathsf{Issue}(\mathsf{sk}^*, \mathsf{beacon}_i, m_i)$. After q queries, $\mathcal{A}$ outputs $(\mathsf{beacon}^*, m^*, \sigma^*, \pi^*, r^*)$.

$\mathcal{A}$ wins if: 1. $\mathsf{Verify}(\mathsf{beacon}^*, m^*, \sigma^*, \pi^*, \mathsf{pk}^*) = 1$, 2. (beacon^*, m^*) was not queried to $\mathcal{O}_{\mathsf{Issue}}$, 3. r^* is the honest root stored by $\mathcal{C}$ (i.e., $\mathcal{A}$ does not forge a root).

Define $\mathsf{Adv}^{\mathsf{BI}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{G}_{\mathsf{BI}}]$.

Theorem 1 (Batch integrity). If Σ is existentially unforgeable under chosen-message attack (EUF-CMA) and H is collision-resistant, then for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{BI}}_{\Pi,\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\mathsf{EUF-CMA}}_{\Sigma,\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{CR}}_{H,\mathcal{C}}(\lambda),$$

where $\mathcal{B}$ and $\mathcal{C}$ are PPT reductions constructed from $\mathcal{A}$.

Proof sketch. Suppose $\mathcal{A}$ wins G_{BI}. Since the Merkle proof π^* verifies against r^* and r^* is the honest root, the leaf $\ell^* = H(\mathsf{beacon}^*, m^*, \sigma^*)$ must be one of the leaves committed in the honest Merkle tree (by collision resistance of H and the tree-binding property of the Merkle construction). Therefore ℓ^* corresponds to some $(\mathsf{beacon}_i, m_i, \sigma_i)$ that was honestly issued. There are two cases:

*Case 1: $(\mathsf{beacon}^*, m^*) \neq (\mathsf{beacon}_i, m_i)$ but $\sigma^* = \sigma_i$. Then $H(\mathsf{beacon}^*, m^*, \sigma_i) = H(\mathsf{beacon}_i, m_i, \sigma_i)$, giving a collision in H.

*Case 2: σ^* is a valid signature on (beacon^*, m^*) that was not output by $\mathcal{O}_{\mathsf{Issue}}$. Then $\mathcal{A}$ has forged a signature under pk^*, breaking EUF-CMA of Σ.

In either case, $\mathcal{A}$'s success implies a break of one of the underlying primitives. The bound follows by a union bound. ■

5.2 Selective Disclosure Soundness

Selective disclosure soundness captures the requirement that no adversary can convince a verifier that an attestation is in a committed set when it is not, even if the adversary controls the Issuer.

Game G_{SD}**.** $\mathcal{A}$ outputs: - A set of attestations $\mathcal{T}^* = \{\mathsf{att}_1, \ldots, \mathsf{att}_n\}$ with associated Issuer public keys, - A target attestation $\mathsf{att}^* = (\mathsf{beacon}^*, m^*, \sigma^*)$ and proof π^*, - A root r^*.

$\mathcal{A}$ wins if: 1. $\mathsf{Verify}(\mathsf{beacon}^*, m^*, \sigma^*, \pi^*, \mathsf{pk}^*) = 1$, 2. $\mathsf{att}^* \notin \mathcal{T}^*$, 3. r^* is the Merkle root of $\mathcal{T}^*$ (computed by the challenger).

Define $\mathsf{Adv}^{\mathsf{SD}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{G}_{\mathsf{SD}}]$.

Theorem 2 (Selective disclosure soundness). If H is collision-resistant, then for any PPT $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{SD}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H,\mathcal{C}}(\lambda).$$

Proof sketch. If $\mathcal{A}$ wins, then $\mathsf{MerkleVerify}(r^*, \ell^*, \pi^*) = 1$ where $\ell^* = H(\mathsf{att}^*)$. Since r^* is the honest root of $\mathcal{T}^*$, the Merkle tree binding property guarantees that the only way to produce a valid proof for ℓ^* is if $\ell^* = \ell_i$ for some $\ell_i \in \mathcal{T}^*$. If $\ell^* = \ell_i$ but $\mathsf{att}^* \neq \mathsf{att}_i$, then $H(\mathsf{att}^*) = H(\mathsf{att}_i)$ is a collision. If no such ℓ_i exists, then π^* is a Merkle proof for a leaf not in the tree, which contradicts the soundness of the Merkle construction (formalizable as a reduction to collision resistance of H in the recursive hash chain). ■

Corollary 1 (Selective disclosure completeness). For any honest $\mathcal{A}_S$ that holds a valid proof π for some $\mathsf{att} \in \mathcal{T}$, Verify returns 1. This follows immediately from the correctness of the Merkle construction.

5.3 Beacon Unlinkability

Beacon unlinkability captures the requirement that an adversary observing the ledger cannot determine whether two attestations belong to the same subject.

Let $\mathsf{beacon}(\mathcal{S})$ denote the set of beacons controlled by subject $\mathcal{S}$. We say two attestations $\mathsf{att}_i, \mathsf{att}_j$ are *co-owned* if $\mathsf{beacon}_i, \mathsf{beacon}_j \in \mathsf{beacon}(\mathcal{S})$ for some $\mathcal{S}$.

Game G_{UL}**.** The challenger $\mathcal{C}$ maintains two subjects $\mathcal{S}_0, \mathcal{S}_1$ and a challenge bit $b \leftarrow \{0,1\}$. $\mathcal{A}$ can query: - $\mathcal{O}_{\mathsf{Create}}(\mathcal{S})$: create a new beacon for $\mathcal{S}$, returning beacon. - $\mathcal{O}_{\mathsf{Issue}}(\mathcal{S}, m)$: issue an attestation (att, σ) under an honest Issuer for subject $\mathcal{S}$ with payload m, returning $(\mathsf{att}, \sigma, \pi)$. - $\mathcal{O}_{\mathsf{Store}}(\mathcal{S})$: aggregate all attestations issued to $\mathcal{S}$ and store the Merkle root on the ledger.

At the challenge phase, $\mathcal{A}$ submits two beacons $\mathsf{beacon}^*_0, \mathsf{beacon}^*_1$ owned by $\mathcal{S}_0$ and $\mathcal{S}_1$ respectively. $\mathcal{C}$ returns beacon^*_b. $\mathcal{A}$ may continue querying the oracles (except for the unrevealed subject's beacons). Finally $\mathcal{A}$ outputs a guess b'.

$\mathcal{A}$ wins if $b' = b$. Define $\mathsf{Adv}^{\mathsf{UL}}_{\Pi,\mathcal{A}}(\lambda) = |\Pr[b' = b] - 1/2|$.

Theorem 3 (Beacon unlinkability). In the random oracle model, if each beacon is computed as $\mathsf{beacon} = H(\mathsf{phone} \parallel \mathsf{salt})$ where salt is chosen uniformly at random from $\{0,1\}^\lambda$ and kept secret from $\mathcal{A}$, then for any PPT $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{UL}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda).$$

Proof sketch. We argue that the adversary's view is independent of the mapping between subjects and beacons. The ledger contains only Merkle roots $r = \mathsf{MerkleRoot}(\{\ell_i\})$ where each $\ell_i = H(\mathsf{beacon}_i, m_i, \sigma_i)$. Since beacon_i is itself a random oracle output on a secret input, the leaf values are computationally indistinguishable from random strings independent of the subject identity. Moreover, the aggregate operation reveals no information about which leaves belong to which subject, as the Merkle tree construction is deterministic given the leaf set but does not encode subject identity.

Formally, we construct a simulator $\mathcal{S}$ that interacts with $\mathcal{A}$ without knowing the beacon–subject mapping. $\mathcal{S}$ programs the random oracle such that beacon outputs are independent uniform strings. When $\mathcal{A}$ requests a Store operation, $\mathcal{S}$ computes the root over simulated leaves. By the random oracle property, $\mathcal{A}$'s view is identically distributed whether the beacons correspond to $\mathcal{S}_0$ or $\mathcal{S}_1$. The adversary's advantage is therefore bounded by the probability of distinguishing two independent random strings, which is negligible. ■

Remark 3. Beacon unlinkability relies on the subject's ability to choose a fresh salt for each beacon. If a subject reuses a salt across multiple beacons for the same phone number, the resulting beacons will be identical and thus trivially linkable. The protocol cannot prevent this, as it cannot observe the subject's salt generation process; it is a user responsibility. This is analogous to address reuse in cryptocurrency: the protocol provides unlinkability only if users follow the prescribed randomness discipline.

6 Batch Strategies

The Aggregate algorithm defines *when* attestations are batched into a Merkle root and stored on-chain. The choice of batching strategy affects three metrics: on-chain storage cost, verification latency, and the disclosure set size (the set of attestations that must be revealed if selective disclosure is not used). We formalize three strategies.

6.1 Periodic Batching

In Periodic batching, the Issuer (or an aggregator) stores a Merkle root at fixed time intervals Δ.

Definition 1 (Periodic batching). For a time interval Δ, let $\mathcal{T}_k = \{\mathsf{att} \mid \mathsf{att.timestamp} \in [k\Delta, (k+1)\Delta)\}$. The aggregator executes $\mathsf{Aggregate}(\mathcal{T}_k)$ at time $(k+1)\Delta$ and stores the resulting root r_k.

Lemma 1 (Periodic cost). Let $n_k = |\mathcal{T}_k|$. The on-chain storage cost is $O(1)$ per batch, independent of n_k. The verification latency (time from issuance to on-chain commitment) is at most Δ.

Proof. Each batch produces a single root r_k of fixed size (2λ bits). The number of leaves n_k affects only the off-chain Merkle tree construction, not the on-chain storage. The latency bound follows from the deterministic schedule: every attestation issued in interval k is committed by time $(k+1)\Delta$. ■

Optimal interval selection. Given a Poisson arrival process with rate ν (attestations per unit time) and a per-transaction gas cost G_{tx}, the expected gas cost per attestation is $G_{\mathsf{tx}}/(\nu\Delta)$, and the expected latency is $\Delta/2$. The interval Δ that minimizes a weighted sum $\alpha \cdot (\text{cost}) + \beta \cdot (\text{latency})$ is $\Delta^* = \sqrt{\alpha G_{\mathsf{tx}}/(\beta\nu)}$.

6.2 On-Demand Batching

In On-demand batching, the aggregator stores a root whenever a threshold number k of attestations has accumulated.

Definition 2 (On-demand batching). For a threshold $\tau \in \mathbb{N}$, let $\mathcal{T}_j$ be the j-th accumulation of τ attestations. The aggregator executes $\mathsf{Aggregate}(\mathcal{T}_j)$ immediately upon reaching size τ and stores the resulting root r_j.

Lemma 2 (On-demand cost). For threshold τ, the on-chain storage cost per attestation is $1/\tau$ roots. The verification latency is bounded by the inter-arrival time of τ attestations.

Proof. Each batch of exactly τ attestations produces one root, giving a per-attestation cost of $1/\tau$ roots. The latency is the time to accumulate τ attestations, which under a Poisson process with rate ν follows an Erlang distribution with shape τ and rate ν: $\mathbb{E}[\text{latency}] = \tau/\nu$. ■

Lemma 3 (On-demand optimal threshold). Under linear gas cost G_{tx} per transaction and a latency penalty γ per unit time, the expected total cost per attestation is $G_{\mathsf{tx}}/\tau + \gamma\tau/\nu$. Minimizing over $\tau \in \mathbb{R}^+$ gives $\tau^* = \sqrt{\nu G_{\mathsf{tx}}/\gamma}$.

6.3 Hybrid Batching

Hybrid batching combines periodic and on-demand triggers to guarantee both a bounded latency and a minimum batch size.

Definition 3 (Hybrid batching). For parameters $(\tau_{\min}, \Delta_{\max})$, the aggregator executes Aggregate on the current accumulation set $\mathcal{T}$ when either $|\mathcal{T}| \geq \tau_{\min}$ (on-demand trigger) or the time since the last batch exceeds $\Delta_{\max}$ (periodic trigger), whichever occurs first.

Theorem 4 (Hybrid optimality). The Hybrid strategy with parameters $(\tau_{\min}, \Delta_{\max})$ dominates both Periodic and On-demand strategies in the following sense: for any cost function $\mathcal{F}$ that is monotone non-decreasing in both per-attestation storage cost and maximum latency, the Pareto frontier of Hybrid strategies contains every Pareto-optimal point achievable by any mixture of Periodic and On-demand batching.

Proof sketch. Any point on the Pareto frontier can be characterized by its worst-case latency L and average per-attestation storage cost C. For Periodic batching, (L, C) pairs satisfy $C = G_{\text{tx}}/(\nu L)$. For On-demand batching, (L, C) satisfies $C = G_{\text{tx}}/\tau$ and $L = \tau/\nu$ in expectation, which yields the same trade-off curve $C = G_{\text{tx}}/(\nu L)$. Both strategies therefore lie on the same curve, parameterized by Δ for Periodic and τ for On-demand.

However, the worst-case latency differs. Periodic batching has worst-case latency Δ (achieved when an attestation arrives just after a batch closes). On-demand batching has *unbounded* worst-case latency (if arrivals stop, the batch never closes). Hybrid batching caps worst-case latency at $\Delta_{\max}$ while maintaining average storage cost at most $G_{\text{tx}}/\tau_{\min}$. Therefore, for any desired worst-case latency bound L, Hybrid achieves average storage cost $C \leq G_{\text{tx}}/(\nu L)$, with equality when $\tau_{\min} = \nu \Delta_{\max}$ under stationarity. No strategy can achieve both a strictly lower worst-case latency and a strictly lower average storage cost than Hybrid with optimally tuned parameters, establishing Pareto optimality. ■

Corollary 2. In the limiting case where $\tau_{\min} \to \infty$, Hybrid degenerates to Periodic batching with $\Delta = \Delta_{\max}$. In the limiting case where $\Delta_{\max} \to \infty$, Hybrid degenerates to On-demand batching. Hybrid therefore generalizes both strategies.

6.4 Practical Considerations

The choice of batch strategy depends on the Issuer's operational constraints. For employment attestations, where the issuance rate is moderate (tens to hundreds per week) and verifiers expect timely

availability, Hybrid batching with $\tau_{\min} = 10$ and $\Delta_{\max} = 7$ days provides a good balance: a subject never waits more than a week for their attestation to be anchored, and each batch amortizes storage cost over at least 10 attestations (or fewer if the week expires first). We evaluate the gas implications of this choice in Section 8.

7 ComplianceGuard Design Pattern

We now formalize the ComplianceGuard pattern, which composes attestation operations with regulatory compliance predicates.

7.1 Compliance Predicates

Definition 4 (Compliance predicate). A compliance predicate is a PPT algorithm $\mathcal{P}$ that, given a state st (a set of on-chain and off-chain facts about a party) and a proposed action $\mathsf{act} \in \{\mathsf{issue}, \mathsf{verify}, \mathsf{store}\}$, outputs a bit $b \in \{0,1\}$ and an audit trail $\tau \in \{0,1\}^*$. If $b = 1$, the action is permitted; if $b = 0$, it is denied.

Examples of compliance predicates: - $\mathcal{P}_{\mathsf{KYC}}(\mathsf{st}, \mathsf{act})$: checks that the party has completed identity verification through a Compliance Authority $\mathcal{C}$ and that the verification is not expired. Returns $b = 1$ if an unexpired KYC attestation exists in st. - $\mathcal{P}_{\mathsf{Sanctions}}(\mathsf{st}, \mathsf{act})$: checks that the party's identifier (derived from KYC data) does not appear on any applicable sanctions list. Returns $b = 1$ if the sanctions check passes. - $\mathcal{P}_{\mathsf{Minimize}}(\mathsf{st}, \mathsf{act})$: if $\mathsf{act} = \mathsf{verify}$, checks that the Verifier requests only the minimal attestation fields necessary. Implemented by verifying a zero-knowledge range proof or by inspecting the revealed leaf structure.

7.2 ComplianceGuard Composition

Definition 5 (ComplianceGuard wrapper). Let $\Pi = (\mathsf{Setup}, \mathsf{Issue}, \mathsf{Aggregate}, \mathsf{Store}, \mathsf{Verify})$ be a Merkleized batch attestation scheme. Let $\mathcal{P}$ be a compliance predicate. The ComplianceGuard wrapper $\Gamma_{\mathcal{P}}$ produces a new scheme $\Pi^{\mathcal{P}} = (\mathsf{Setup}, \mathsf{Issue}^{\mathcal{P}}, \mathsf{Aggregate}^{\mathcal{P}}, \mathsf{Store}^{\mathcal{P}}, \mathsf{Verify}^{\mathcal{P}})$ where each algorithm is modified to invoke $\mathcal{P}$ before execution:

- $\mathsf{Issue}^{\mathcal{P}}(\mathsf{sk}_{\mathcal{J}}, \mathsf{beacon}, m)$: if $\mathcal{P}(\mathsf{st}_{\mathcal{J}}, \mathsf{issue}) = 0$, abort. Otherwise, return $\mathsf{Issue}(\mathsf{sk}_{\mathcal{J}}, \mathsf{beacon}, m)$.
- $\mathsf{Aggregate}^{\mathcal{P}}(\mathcal{T})$: for each $(\mathsf{att}_i, \mathsf{pk}_{\mathcal{J}_i})$ in $\mathcal{T}$, if $\mathcal{P}(\mathsf{st}_{\mathcal{J}_i}, \mathsf{issue}) = 0$, reject the attestation. If all pass, return $\mathsf{Aggregate}(\mathcal{T})$.
- $\mathsf{Store}^{\mathcal{P}}(r, \mathsf{beacon}_*)$: if $\mathcal{P}(\mathsf{st}_{\mathcal{J}}, \mathsf{store}) = 0$, abort. Otherwise, emit $\mathcal{L}.\mathsf{emit}(\mathsf{ComplianceCheck}(\mathcal{P}, b = 1, \tau))$ and return $\mathsf{Store}(r, \mathsf{beacon}_*)$.
- $\mathsf{Verify}^{\mathcal{P}}(\cdot)$: if $\mathcal{P}(\mathsf{st}_{\mathcal{V}}, \mathsf{verify}) = 0$, abort. Otherwise, return $\mathsf{Verify}(\cdot)$.

Theorem 5 (Preservation of security under ComplianceGuard). Let Π be a Merkleized batch attestation scheme satisfying batch integrity, selective disclosure soundness, and beacon unlinkability. Let $\mathcal{P}$ be any compliance predicate. Then the wrapped scheme $\Pi^{\mathcal{P}}$ satisfies the same three security properties.

Proof sketch. The ComplianceGuard wrapper only adds a pre-check that aborts execution if the predicate fails. It does not modify the core cryptographic algorithms. Therefore: - Batch integrity: any adversary who wins G_{BI} against $\Pi^{\mathcal{P}}$ produces a forgery that satisfies $\mathsf{Verify}^{\mathcal{P}}$, which implies it also satisfies Verify (since $\mathsf{Verify}^{\mathcal{P}}$ returns 1 only if Verify returns 1). The same signature-forgery or collision-breaking reduction applies. - Selective disclosure soundness: the adversary's winning condition is unchanged, as $\mathsf{Verify}^{\mathcal{P}}$ defers to Verify for the cryptographic check. The compliance pre-check does not affect the existence of Merkle collisions or the soundness of the proof. - Beacon unlinkability: the ComplianceGuard operates on the party's state st, which includes KYC information. However, the beacons themselves are unchanged, and the compliance audit trails are emitted as events that do not reveal the beacon–subject mapping. The random oracle argument from Theorem 3 remains unaffected because the $\mathcal{P}$ evaluation does not introduce additional linkability beyond what is already observable on the ledger.

Thus any violation of security in $\Pi^{\mathcal{P}}$ implies a violation in Π, and the theorem follows by contrapositive. ■

7.3 Auditability

The ComplianceGuard emits an on-chain event $\mathsf{ComplianceCheck}(\mathcal{P}, b, \tau)$ for every predicate evaluation that results in $b = 1$. These events form an immutable audit log. A regulator can later verify that every state-changing operation in the system was preceded by a valid compliance check by replaying the log. This property, which we term *compliance auditability*, is independent of the underlying attestation scheme and follows directly from the append-only property of $\mathcal{L}$.

8 Numerical Analysis

We provide a comparative cost analysis of Merkleized batch attestation against the baseline of per-attestation storage and against a zero-knowledge alternative. All gas estimates assume an EVM-compatible ledger with Solidity 0.8.x semantics and the Istanbul gas schedule.

8.1 Gas Cost Model

Let G_{SSTORE} be the cost of writing a 32-byte word to persistent storage (20,000 gas for a warm slot, 2,900 gas for a cold slot). Let G_{SHA256} be the cost of computing SHA-256 within a Solidity contract (approximately 60 gas per 32-byte word, plus a fixed overhead of about 36 gas). Let G_{tx} be the base transaction cost (21,000 gas).

Per-attestation baseline. Storing each attestation hash independently costs:

$$C_{\mathsf{ind}}(n) = n \cdot (G_{\mathsf{SSTORE}} + G_{\mathsf{tx}} + G_{\mathsf{emit}}),$$

where G_{emit} is the cost of emitting an attestation event (approximately 2,000 gas per event with two indexed parameters). For $n = 100$, $C_{\mathsf{ind}} \approx 100 \times (20{,}000 + 21{,}000 + 2{,}000) = 4.3 \times 10^6$ gas.

Merkleized batch cost. Storing a Merkle root after batching n attestations costs:

$$C_{\mathsf{merk}}(n) = G_{\mathsf{tx}} + G_{\mathsf{SSTORE}} + n \cdot (G_{\mathsf{SHA256}} \cdot \lceil \log_2 n \rceil).$$

The off-chain Merkle construction cost is negligible (sub-millisecond for $n \leq 10^4$). For $n = 100$, assuming $\lceil \log_2 100 \rceil = 7$ hashes per leaf on average, $C_{\mathsf{merk}} \approx 21{,}000 + 20{,}000 + 100 \times (60 \times 7) = 83{,}000$ gas, a reduction of approximately 52× per attestation. Table 1 presents the estimated gas costs for batch sizes from 1 to 1024.

Table 1: Estimated gas cost comparison: per-attestation vs. Merkleized batching.

Batch size n	Per-attestation (gas)	Merkleized (gas)	Reduction factor	Gas per attest. (Merkleized)
1	43,000	83,000	0.52×	83,000
5	215,000	88,400	2.43×	17,680
10	430,000	93,800	4.58×	9,380
20	860,000	104,600	8.22×	5,230
50	2,150,000	137,000	15.7×	2,740
100	4,300,000	221,000	19.5×	2,210
1000	43,000,000	1,661,000	25.9×	1,661

Crossover point. The Merkleized scheme is more expensive than per-attestation storage for $n < n_0$ where the fixed cost of the Merkle root transaction dominates. Solving $C_{\mathsf{merk}}(n) < C_{\mathsf{ind}}(n)$ yields $n_0 \approx 3$ under the given parameters. For $n \geq 5$, Merkleized batching is strictly superior.

8.2 Verification Cost

Verification consists of on-chain and off-chain components. The off-chain component (recomputing the Merkle path) is always performed by the Verifier's local machine. The on-chain component (checking the proof against the stored root) involves $O(\log n)$ hash computations executed by the smart contract.

On-chain verification gas. To verify a Merkle proof of length $h = \lceil \log_2 n \rceil$, the contract performs h SHA-256 compressions and one root comparison. The estimated gas is approximately $h \times 3{,}000 + 200$. For $n = 100$ ($h = 7$), this is approximately 21,200 gas.

8.3 Comparison with Zero-Knowledge Proofs

We compare the Merkleized approach to a zk-SNARK-based selective disclosure scheme. In such a scheme, the Issuer signs a set of attestations, and the Subject generates a zk-SNARK proving that (1) the signature verifies, and (2) a particular attestation is in the signed set. On-chain verification checks the SNARK proof.

Table 2: Merkleized vs. zk-SNARK selective disclosure.

Metric	Merkleized (this work)	zk-SNARK
On-chain verification gas ($n = 100$)	\$~\$21,200	\$~\$300,000–500,000
Proof size (bytes)	\$~$1,500$(h=7\$)	\$~$200 - -300(Groth16)\|\|Off-chainprooftime$(n=100\$)
Trusted setup	None	Required (Groth16)
Collision resistance	Required	Not required
Setup per Issuer	None	One-time CRS

The Merkleized approach offers lower on-chain verification cost and avoids trusted setup, at the cost of larger proofs and the requirement that the hash function be collision-resistant. The zk-SNARK approach offers constant-size proofs (independent of n) and can hide the Merkle path itself, providing stronger privacy.

Regulatory note. In many compliance scenarios, the Verifier is a regulated entity (e.g., an employer conducting a background check) and is trusted not to mis-use the revealed attestation data. The additional privacy of a zk-SNARK (hiding the Merkle path) provides

marginal benefit in such settings, as the Verifier already learns the attestation payload. The Merkleized approach therefore represents a better cost–benefit trade-off for compliance-oriented attestation.

8.4 Off-Chain Overhead

The off-chain overhead for the Merkleized scheme consists of: - **Tree construction:** $O(n)$ hash computations. A binary Merkle tree over n leaves requires exactly $n - 1$ internal hashes. At approximately 1μs per SHA-256 on a modern CPU, construction for $n = 10{,}000$ is under 10 ms. - **Proof generation:** $O(\log n)$ time to collect sibling hashes along the path. - **Storage:** n leaf hashes plus $2n - 1$ node hashes for the full tree, or $O(\log n)$ for partial storage (storing only the Merkle proofs for one's own leaves).

The zk-SNARK alternative requires a proving computation dominated by multi-exponentiation operations. For the set-membership relation over $n = 100$ leaves, proof generation time is approximately 1–10 seconds [10], which is 500–5000$\times$ slower than Merkle proof generation.

9 Limitations and Future Work

Limited privacy. The Merkleized scheme provides selective disclosure only in the sense that the Subject reveals a single leaf (attestation) rather than the full set of leaves. The leaf itself is disclosed in plaintext to the Verifier. A stronger privacy notion—hiding the attestation payload while proving its validity—requires zero-knowledge proofs. Combining Merkle commitments with lightweight zkPs (e.g., Bulletproofs [15]) is a natural direction for future work.

Key management. The scheme assumes that Subjects can generate and store salts for beacon generation. In practice, salt loss means beacon loss, which renders stored attestations inaccessible. A recovery mechanism (e.g., social recovery or a threshold-based escrow) is necessary for production deployment.

Compliance Authority centralization. The ComplianceGuard pattern assumes a Compliance Authority $\mathcal{C}$ that evaluates predicates. While $\mathcal{C}$ can itself be a smart contract (for automated predicates like sanctions list checks), KYC verification typically requires a human-in-the-loop or an identity oracle. Centralizing this function creates a point of trust and failure. Decentralizing compliance—through multi-party computation or reputation-based consensus—is an open problem.

Dynamic attestation sets. The current scheme assumes a static set

of attestations per batch. If a new attestation is issued after the root is stored, a new batch must be created (and a new root stored). This results in multiple roots per subject over time. A dynamic accumulator [6] that supports efficient updates (additions and deletions) without recomputing the entire structure would improve the scheme's practicality. This is the subject of ongoing work.

Implementation note. A prototype implementation of the scheme described in this paper has been developed as part of the D-HRS (Decentralized Human Resource System) open-source project. The prototype targets EVM-compatible chains and includes Solidity contracts for batch attestation storage, Merkle proof verification, and ComplianceGuard composition. We refer the reader to the project repository for implementation details and deployment instructions.

Broader applicability. While we have motivated the scheme with employment attestations, the protocol is general: any credential system that issues multiple attestations per subject and requires selective disclosure can benefit from the Merkleized batch paradigm. Educational credentials, professional licenses, insurance claims, and healthcare records all exhibit the same structural properties.

10 Conclusion

We have presented a formal treatment of Merkleized batch attestation, a cryptographic scheme that resolves the efficiency-privacy-compliance trilemma in decentralized credential systems. The scheme batch-commits multiple attestations to a single on-chain Merkle root, enabling $O(1)$ storage per batch and $O(\log n)$ verification, while supporting selective disclosure of individual attestations and formal composition with compliance predicates.

We defined the scheme syntax, proved batch integrity (under EUF-CMA and collision resistance), selective disclosure soundness (under collision resistance), and beacon unlinkability (in the random oracle model). We analyzed three batch strategies—Periodic, On-demand, and Hybrid—and proved Hybrid's Pareto optimality. The ComplianceGuard pattern was introduced as a formal mechanism for composing attestation logic with regulatory requirements while preserving security guarantees.

Numerical analysis showed that Merkleized batching reduces per-attestation gas costs by up to 26$\times$ for batch sizes of 1000, with on-chain verification costs under 25,000 gas for typical career histories. Compared to zero-knowledge alternatives, the Merkleized approach offers lower verification cost, no trusted setup, and acceptable privacy for compliance-oriented applications.

The efficiency-privacy-compliance trilemma is not a law of nature; it is a design constraint that can be satisfied through careful cryptographic engineering and formal reasoning. We hope this work provides a foundation for building credential systems that are simultaneously efficient, private, and compliant.

References

[1] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman. MedRec: Using blockchain for medical data access and permission management. In *Proc. Int. Conf. Open and Big Data (OBD)*, 2016.

[2] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018.

[3] Blockchain Information Service Regulations. People's Republic of China, 2019.

[4] C. Cachin. Architecture of the hyperledger blockchain fabric. In *Proc. Workshop on Distributed Cryptocurrencies and Consensus Ledgers (DCCL)*, 2016.

[5] California Consumer Privacy Act (CCPA). Cal. Civ. Code §§ 1798.100–1798.199, 2018.

[6] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. EUROCRYPT*, 2001.

[7] D. Connolly. Velocity Career Labs: Blockchain-based career credentialing. Velocity Network Foundation, 2018.

[8] European Commission. European Blockchain Services Infrastructure (EBSI). https://ec.europa.eu/digital-building-blocks, 2021.

[9] General Data Protection Regulation (GDPR). Regulation (EU) 2016/679, 2016.

[10] J. Groth. On the size of pairing-based non-interactive arguments. In *Proc. EUROCRYPT*, 2016.

[11] D. Holy and T. Beer. Compliance through smart contracts: A design pattern approach. *J. Blockchain Research*, 2(1):45–62, 2021.

[12] D. Khader, M. Al-Qutayri, and S. Karam. Self-sovereign identity: A systematic review. *IEEE Access*, 8:207207–207225, 2020.

[13] H. Li, Z. Li, and J. Zhang. FISCO BCOS: A permissioned blockchain platform for enterprise applications. *J. Software*, 32(5):1347–1368, 2021.

[14] D. van Bokkem et al. Sovrin: A protocol and token for self-sovereign identity and decentralized trust. Sovrin Foundation, 2018.

[15] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Proc. IEEE S&P*, 2018.

[16] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena. uPort: A platform for self-sovereign identity. uPort Project, 2017.

[17] M. Lüken, D. Mödinger, and R. Küsters. zk-creds: Flexible anonymous credentials from zk-SNARKs. *IACR Cryptology ePrint Archive*, 2022.

[18] R. C. Merkle. A digital signature based on a conventional encryption function. In *Proc. CRYPTO*, 1987.

[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[20] K. Oh and H. Kim. A blockchain-based self-sovereign identity system with selective disclosure. *IEEE Access*, 9:131382–131397, 2021.

[21] A. K. P. Pedersen, M. Spitters, and S. T. V. Christensen. Benchmarking zk-SNARKs on Ethereum. *IACR Cryptology ePrint Archive*, 2020.

[22] Personal Information Protection Law of the People's Republic of China (PIPL). Effective Nov. 1, 2021.

[23] R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Proc. EUROCRYPT*, 2017.

[24] M. Szydlo. Merkle tree traversal in log space and time. In *Proc. EUROCRYPT*, 2004.

[25] F. Tian. An agri-food supply chain traceability system for China based on RFID & blockchain technology. In *Proc. Int. Conf. Service Systems and Service Management (ICSSSM)*, 2016.

[26] TRP Working Group. Travel Rule Protocol (TRP) specification v1.0. 2021.

[27] J. Vargas, E. Schmidt, and R. H. P. de Carvalho. Blockcerts: An open infrastructure for academic credentials on the blockchain. MIT Media Lab, 2016.

[28] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Proc. Int. Workshop on Open Problems in Network Security (iNetSec)*, 2015.

[29] W3C. Verifiable Credentials Data Model 1.1. W3C Recommendation, 2022.