CUBE: Partially Blind BBS Signatures for Unlinkable Decentralized Identity

A Succinct On-Chain Verifiable Credential Framework

Guohao Lai

Jinan University, Guangzhou, China

Abstract

Decentralized Identity (DID) systems aim to provide users with self-sovereign control over their credentials. However, verifying identities on public blockchains often risks leaking sensitive attributes and transaction histories, whereas relying on heavy generic cryptographic protocols leads to prohibitive on-chain costs. In this paper, we propose **CUBE** (Credential Unlinkability using Blinded Elements), a verifiable credential framework that integrates partially blind BBS signatures with Non-Interactive Zero-Knowledge (NIZK) proofs. By shifting complex identity validation off-chain while blinding the core identity-binding elements, the scheme ensures on-chain unlinkability without losing attribute auditability. To prevent Sybil attacks and credential transfer, we integrate a deterministic nullifier tied to Soulbound Tokens (SBTs). We formalize the scheme's system model and establish its security regarding existential unforgeability, signer-blindness, and zero-knowledge. Performance results show that this construction maintains minimal on-chain gas consumption, offering a practical deployment path for privacy-preserving verifiable credentials in Web3 environments.

Keywords: Decentralized Identity, Verifiable Credentials, BBS Signatures, Zero-Knowledge Proofs, Blockchain, Blind Signature

1 Introduction

1.1 Motivation

The transition to Web3 applications has driven digital identity management toward user-centric models. Decentralized Identity (DID) [RS+22] systems and Verifiable Credentials (VC) [SLL+22] enable individuals to manage their digital footprints independently, reducing reliance on centralized identity providers and limiting exposure to data breaches.

Anonymous Credentials (AC) [CL04] serve as a core component for privacy-preserving authentication. The BBS signature scheme, originally proposed by Boneh, Boyen, and Shacham [BBS04] and adapted for modern credentials [CDL16, TZ23], is widely used in this domain. Because BBS signatures natively support selective disclosure and zero-knowledge proofs, they are being integrated into W3C standards and frameworks such as the EU Digital Identity Wallet (eIDAS 2.0).

However, deploying these identity mechanisms on public blockchains presents privacy and scalability challenges. While generic zero-knowledge arguments (e.g., zk-SNARKs [Gro16]), zero-knowledge verifiable encryption (e.g., SAVER [LCKO19]), and auditing extensions [ZLB24]

Email: klizz@ciall0.cn (Guohao Lai)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-05-10.

can provide on-chain privacy, directly applying them to DID verification usually incurs excessively high gas costs. In addition, standard on-chain credentials risk being traded like generic digital assets, defeating the non-transferability requirement of identity attributes. Soulbound Tokens (SBTs) [WOB22] restrict transferability by binding tokens to specific blockchain addresses, but enabling privacy-preserving interactions for SBTs remains computationally challenging.

SBTs provide utility for on-chain identity by enabling applications such as KYC-gated DeFi protocols, reputation systems for decentralized development communities, and role-based access control. They bind authenticated attributes to a user's blockchain address. Adding privacy layers to SBTs allows users to prove these attributes on-chain without explicitly revealing the underlying data.

1.2 Limitations of Prior Work

Existing solutions for unlinkable decentralized identities face significant trade-offs. Recent frameworks like OpenAC [ENR+26] utilize general-purpose zk-SNARKs to preserve issuer key compatibility; however, they suffer from prohibitive on-chain verification costs due to the lack of native EVM precompiles for their verification algorithms, resulting in gas overheads often exceeding 1 million. While pioneering threshold credential systems like Coconut [SAB+19] and recent threshold BBS+ schemes [DKL+23] attempt to distribute trust, they introduce substantial communication complexity during issuance and are primarily designed for off-chain verification, making them inefficient for high-frequency on-chain applications. Furthermore, standard BBS signatures [BBS04] are inherently *linkable*, allowing issuers to track users across different sessions—a critical privacy gap that prior art fails to address without sacrificing efficiency.

1.3 Practical Web3 authentication patterns and limitations

Current implementations link off-chain accounts to on-chain addresses through relatively lightweight methods, such as OAuth-based account linking, social media proofs, Sign-In-With-Ethereum (SIWE), and centralized KYC attestations. While functional, these methods suffer from dependencies on centralized providers, risks of account takeover, permanent on-chain linkability, and vulnerability to Sybil attacks. Conversely, full zero-knowledge solutions remain costly.

CUBE addresses these practical limitations. By utilizing partially blind BBS issuance (hiding address and nullifier data from the Signer) alongside deterministic address-bound nullifiers, CUBE maintains unlinkability between the issuance phase and the on-chain presentation. It enforces a one-time issuance of SBTs and offloads the intensive attribute verification to the off-chain Signer to keep on-chain execution succinct. The design acknowledges certain trade-offs, such as reliance on the Signer for availability (which threshold issuance systems like Coconut [SAB+19] can mitigate) and the additional client-side computation required for blinding.

Figure 1 illustrates the system architecture and the boundary between off-chain identity validation and on-chain issuance.

1.4 Our Contributions

To resolve these challenges, we propose **CUBE**, a framework combining partially blind BBS signatures and address-bound deterministic nullifiers to support anonymous and Sybil-resistant decentralized identity checks on-chain. Our main contributions are:

1. **Decoupled DID Architecture:** We provide a verification model that delegates intensive attribute validation (e.g., KYC processes) to an off-chain issuer. The on-chain smart contracts are then only responsible for succinct cryptographic verification.

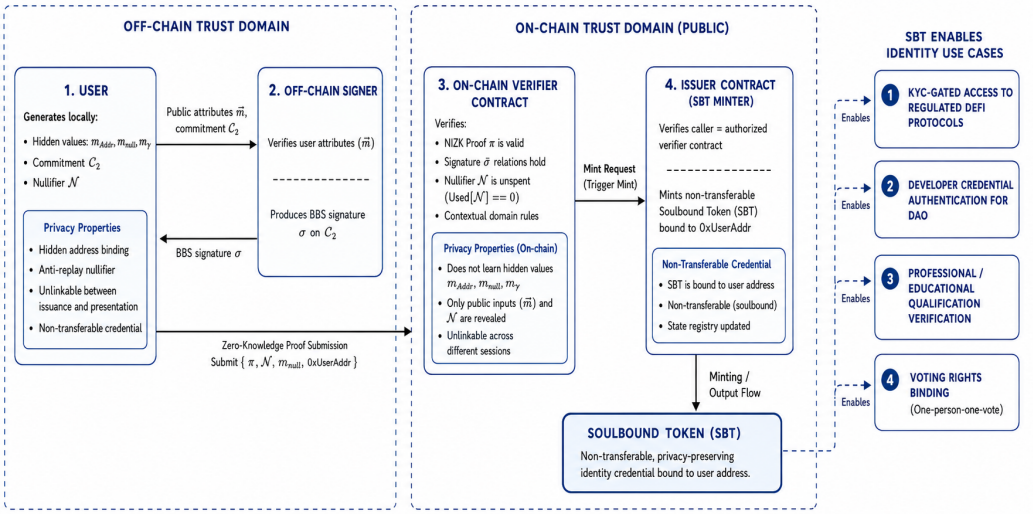


Figure 1: System architecture of our scheme.

2. **On-chain Unlinkability:** We apply a partially blind extension to BBS issuance. By blinding the user's specific identity parameters from the Signer, our scheme breaks the linkability, ensuring that the on-chain NIZK proof verification cannot be correlated.

3. **Sybil Resistance via SBTs:** Constructing upon nullifier concepts from privacy-preserving cryptocurrencies (e.g., Zcash [BSCG+14]) and mixing protocols (e.g., Tornado Cash [PSS19]), we bind zero-knowledge operations to deterministic nullifiers. This approach translates verifiable credentials into privacy-preserving Soulbound Tokens, preventing credential replay and restricting multi-address double-spending.

1.5 Paper Organization

The remainder of this paper is structured as follows. Section 2 reviews the underlying cryptographic primitives, including bilinear pairings and BBS signatures. Section 3 formalizes the system model and security definitions. Section 4 outlines the construction and algorithms of the CUBE scheme. Section 5 covers the security proofs. Section 6 details the performance and gas consumption of our smart contract implementation, and Section 7 concludes the paper.

2 Preliminaries

2.1 Bilinear Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic cryptographic groups of prime order p, and let $\mathbb{G}_T$ be a target cyclic group of the same order p. An asymmetric bilinear pairing is a computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying the following properties:

- **Bilinearity:** For all $U \in \mathbb{G}_1, V \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(aU, bV) = e(U, V)^{ab}$.
- **Non-degeneracy:** If G_1 generates $\mathbb{G}_1$ and G_2 generates $\mathbb{G}_2$, then $e(G_1, G_2)$ generates $\mathbb{G}_T$.

- **Computability:** There exists an efficient algorithm to compute $e(U, V)$ for any $U \in \mathbb{G}_1, V \in \mathbb{G}_2$.

The security of schemes constructed over these groups typically relies on standard assumptions for pairing-friendly curves. Chief among them is the q-Strong Diffie-Hellman (q-SDH) assumption [BB04, TZ23]. Formalized by Boneh and Boyen, the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ posits that given a $(q+2)$-tuple $(G_1, G_2, xG_2, x^2G_2, \ldots, x^qG_2)$ for a uniformly random $x \xleftarrow{\$} \mathbb{Z}_p^*$, it is computationally intractable for any probabilistic polynomial-time adversary to output a pair $(c, \frac{1}{x+c}G_1)$ such that $c \in \mathbb{Z}_p$. This assumption forms the fundamental security basis for both our partially blind parameter extensions and the underlying BBS signature unforgeability.

2.2 BBS Signatures

The BBS signature scheme, originally proposed by Boneh, Boyen, and Shacham, is a short signature scheme designed for signing vectors of messages. A standard BBS signature on a message vector $\vec{m} = \{m_1, \ldots, m_n\}$ consists of a pair $(A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$. The scheme operates over the bilinear pairing parameters with three core algorithms:

- **KeyGen:** The signer samples a random secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and computes the public key $X = x \cdot G_2$. Public parameters include independent generators $H_1, \ldots, H_n \in \mathbb{G}_1$ for mapping messages.
- **Sign:** To sign a message vector $\vec{m} \in \mathbb{Z}_p^n$, the signer samples a random scalar $e \xleftarrow{\$} \mathbb{Z}_p$, and computes $A = \frac{1}{x+e}(G_1 + \sum_{i=1}^{n} m_i H_i)$. The signature is $\sigma = (A, e)$.
- **Verify:** Given the signature $\sigma = (A, e)$, public key X, and messages $\vec{m}$, the verifier checks the pairing relation: $e(A, X + eG_2) = e(G_1 + \sum_{i=1}^{n} m_i H_i, G_2)$.

Recent work by Tessaro and Zhu [TZ23] formalized the security of the core BBS signature, proving its strong unforgeability under the q-SDH assumption in the standard model. This resolved previous theoretical gaps regarding the short form of BBS compared to the BBS+ variant [CDL16], which requires an additional scalar s to form $\sigma = (A, e, s)$. We utilize BBS signatures because they inherently support selective disclosure, allowing a prover to demonstrate possession of a valid signature on $\vec{m}$ while keeping a subset of the messages (e.g., blind factors) hidden [DKL+23, TZ23].

2.3 Zero-Knowledge Proofs & Fiat-Shamir Heuristic

Zero-Knowledge Proofs of Knowledge (ZKPoK) enable a prover to convince a verifier that they possess a secret witness satisfying a mathematical statement, without revealing the witness itself. In our scheme, ZKPoK is used to prove possession of a valid BBS signature and selectively disclose attributes while maintaining zero-knowledge properties [TZ23].

Interactive zero-knowledge protocols can be converted into Non-Interactive Zero-Knowledge (NIZK) proofs using the Fiat-Shamir heuristic [FS87]. By deriving the verifier's challenge c via a collision-resistant hash function applied to the public statement (context, revealed messages, system parameters) and the prover's commitments, the protocol becomes non-interactive. This transformation makes the proofs publicly verifiable and suitable for on-chain smart contract execution.

2.4 Soulbound Tokens (SBT)

Soulbound Tokens (SBTs) are non-transferable, identity-bound digital tokens. SBTs link explicit attestations or verifiable credentials directly to specific blockchain addresses.

Through standardized interfaces such as ERC-5114 or ERC-5484, smart contracts enforce transfer restrictions, tying the token permanently to an owner. In our architecture, SBTs serve as the on-chain representation of off-chain verified credentials, allowing decentralized applications to verify identity properties directly without repeating the underlying zero-knowledge proof validations.

3 System Model and Security Definitions

3.1 System Entities

Our proposed CUBE framework consists of four primary entities, designed to decouple intensive cryptographic computations from on-chain execution:

- **User (Prover)**: A participant requesting a verifiable credential (e.g., a decentralized passport) to authorize on-chain interactions. The User interacts with the Signer off-chain to obtain the credential and generates a Non-Interactive Zero-Knowledge (NIZK) proof to authenticate anonymously on-chain.
- **Signer (Trusted Third Party / TTP)**: An off-chain authorized node holding a private signing key. It is responsible for verifying user attributes and issuing BBS signatures over the user's partially hidden commitments.
- **Verifier (Relayer Contract)**: An intermediary smart contract that performs on-chain authentication of the NIZK proofs and triggers callback functions.
- **Issuer (DApp Contract)**: An on-chain target contract that manages the minting of address-bound SBTs upon successful verification.

3.2 Algorithms and Syntax

The CUBE protocol is formalized as a tuple of five polynomial-time algorithms $\Pi = (\mathsf{Init}, \mathsf{Apply}, \mathsf{Sign}, \mathsf{Prove}, \mathsf{Verify})$:

- $\mathsf{Init}(1^\lambda) \to (\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$: Executed by the Issuer and Signer. Takes the security parameter 1^λ and outputs public parameters pp (including pairing groups and standard generators), the Signer's secret key sk, and the corresponding public key pk.
- $\mathsf{Apply}(\mathsf{pp}, \mathsf{addr}, \vec{m}) \to (\mathcal{C}_2, \mathsf{st}_{\text{user}})$: Executed by the User. Takes the user's blockchain address addr and attribute vector $\vec{m}$. It generates random scalars including an ephemeral blinding factor, computes a foundational commitment $\mathcal{C}_2$, and records an internal state $\mathsf{st}_{\text{user}}$.
- $\mathsf{Sign}(\mathsf{sk}, \mathcal{C}_2, \vec{m}) \to (\sigma, \perp)$: Executed by the Signer. Verifies the public attributes in $\vec{m}$. If approved, it computes the public commitment segment, aggregates it with $\mathcal{C}_2$, and signs the result using sk to produce a signature σ. Otherwise, it outputs $\perp$.
- $\mathsf{Prove}(\mathsf{pp}, \mathsf{pk}, \sigma, \mathsf{st}_{\text{user}}, M) \to \pi$: Executed by the User. Uses the internal state and the received signature to generate a verifiable Non-Interactive Zero-Knowledge (NIZK) proof π, randomizing the signature to preserve zero-knowledge, bound to an optional context message M. Outputs the proof payload including π, the nullifier seed, and the user's address.
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, \pi, M, m_{\text{null}}, \mathsf{addr}) \to \{0, 1\}$: Executed by the on-chain Verifier. Reconstructs the public commitment segment, computes the nullifier $\mathcal{N}$, and validates the proof π with respect to the context M and structural algebraic consistency. Returns 1 (accept) to trigger SBT minting, or 0 (reject).

3.3 Security Requirements

A secure Decentralized Identity scheme operating over public blockchains must satisfy four primary security properties:

3.3.1 Existential Unforgeability against Chosen Message Attacks (EUF-CMA)

No Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$, even with access to a signing oracle $\mathcal{O}_{\text{Sign}}$, should be able to produce a valid NIZK proof π^* for an attribute vector $\vec{m}^*$ that was not previously authorized by the Sign procedure. This prevents the fabrication of valid credentials.

3.3.2 Signer-Blindness and On-chain Unlinkability

The verification process must not leak identifying correlations. Even a malicious Signer possessing all historical blinded queries $\mathcal{C}_2$ must be unable to link a valid on-chain proof π to its corresponding issuance transcript with a probability non-negligibly better than random guessing.

3.3.3 Sybil Resistance and Non-Transferability (Double-Spending Prevention)

Leveraging the deterministically bound nullifier $\mathcal{N}$, the scheme must prevent credential replay or token double-spending. Using a single authorized credential $\mathsf{st}_{\text{user}}$ to authenticate multiple distinct blockchain addresses must be computationally equivalent to solving the Discrete Logarithm (DL) problem, thus enforcing a strict 1-to-1 token minting constraint.

3.3.4 Zero-Knowledge Capability

The public NIZK proof π published on-chain must reveal no information about the underlying attributes beyond the fact that they satisfy the required conditions. Observers learn nothing other than the binary validity evaluation of the Verify function.

4 Proposed Scheme

4.1 Design Rationale

Our scheme's architecture is developed to balance security, user privacy, and on-chain efficiency.Figure 2 gives an overview of the end-to-end protocol flow, including off-chain signing, proof generation, and on-chain verification. The design rationale centers on the following objectives:

1. **Decoupled Verification to Minimize On-chain Footprint:** To avoid the high gas costs of on-chain data processing, complex identity validations (e.g., KYC) are performed by an off-chain Trusted Third Party (*Signer*). The blockchain smart contracts (*Verifier*/*Issuer*) are restricted to evaluating succinct zero-knowledge proofs and minting the resulting identity-bound Soulbound Tokens (SBTs).

2. **Cryptographic Unforgeability via BBS Signatures:** Credential authenticity relies on the unforgeability of BBS signatures. Only the authorized Signer, holding the secret key x, can compute a valid signature over the attributes, preventing credential fabrication.

3. **Sybil Resistance and Anti-Replay via Bound Nullifiers:** To prevent sharing and replay attacks, each credential is bound to the user's destination blockchain

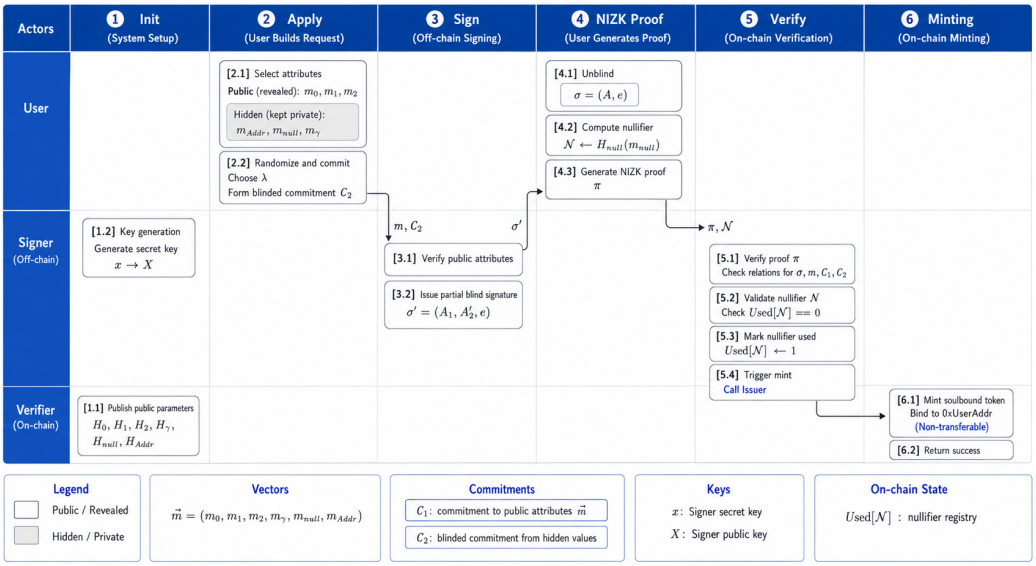


Figure 2: Protocol flow of our scheme from initialization and partially blind signing to NIZK verification and SBT minting.

address (m_{Addr}). This is enforced through a deterministic nullifier:

$$\mathcal{N} = \text{Hash}(0x\text{UserAddr} \parallel m_{\text{null}})$$

By recording $\mathcal{N}$ on-chain, the protocol establishes a strict one-to-one mapping between a signature and a minted credential, requiring users to request a new signature for subsequent issuances.

4. **Mitigating Signer Traceability via Commitments:** If the user transmits the nullifier seed (m_{null}) and destination address (m_{Addr}) in plaintext, the Signer learns the user's on-chain identity. To prevent this, the protocol hides these identity-binding components within a cryptographic commitment using an ephemeral scalar m_γ as randomness.

5. **Addressing Linkability in Standard Commitments:** Committing to the variables is insufficient for unlinkability if the Signer issues a standard signature directly over the commitment tuple. The algebraic structure would remain deterministic, allowing the Signer to correlate the on-chain zero-knowledge proof (π) with the initial issuance transcript.

6. **Achieving Anonymity with Partially Blind Signatures:** To break this linkage, we use a *partial blind signature* mechanism. The user hides the identity-binding commitment with an ephemeral randomizer m_γ during the signing request. The identity-binding and anti-Sybil components ($m_{\text{Addr}}, m_{\text{null}}, m_\gamma$) are concealed from the Signer, while the explicit attributes ($m_0, m_1, \dots$) remain in plaintext. This allows the Signer to validate the factual claims prior to signing while providing the user with unlinkability during on-chain execution.

4.2 System Architecture

- **User**: The participant seeking a verifiable credential (e.g., a decentralized passport) to authorize on-chain interactions. The user registers a USERID (e.g., email, phone

number) with the Signer.

- **Issuer**: An on-chain smart contract deployed by the DApp, responsible for minting an address-bound soulbound token (SBT, complying with ERC-5114 or ERC-5484) to the verified User.
- **Signer**: An off-chain Trusted Third Party (TTP) node that evaluates user identity attributes and issues valid BBS signatures.
- **Verifier**: An intermediary contract that authenticates the user's NIZK proofs and triggers automated callbacks to the Issuer contract to execute minting protocols.

4.3 Symbol Definitions

- $< \cdots >$: Denotes a relation of the data in the braces.
- $0x$UserAddr (mapped to scalar m_{Addr}): The user's address on the blockchain.
- $\mathcal{N}$: The nullifier, a unique identifier derived from the user's address and a random seed, used to prevent double-spending of credentials.
- $\vec{m}$: A vector of public message components representing explicit identity constraints.
- ctx: A context string used in the Fiat-Shamir transformation to bind the proof to specific parameters and prevent replay attacks (e.g. chainID, name of the dapp).
- TokenID: The final identifier of the minted SBT, deterministically derived from the nullifier $\mathcal{N}$ to ensure a strict one-to-one mapping between credentials and tokens.

4.4 Message Coding Definitions

To mathematically formalize the constraints, we designate specific properties to unique discrete message generators:

Variable	Visibility	Message Contents / Identity Constraints
m_0, H_0	ALL	Explicit Constraint 1 (e.g., `Hash("Is_Adult=true")`)
m_1, H_1	ALL	Explicit Constraint 2 (e.g., `Hash("Nationality=US")`)
m_2, H_2	ALL	Explicit Constraint 3 (e.g., `Hash("TelephoneVerified=true")`)
$\cdots$	$\cdots$	$\cdots$
$m_{\ell-1}, H_{\ell-1}$	ALL	Explicit Constraint ℓ
m_γ, H_γ	Only User	Ephemeral Blinding Factor (Concealed from the Signer)
$m_{\text{null}}, H_{\text{null}}$	Except Signer	Nullifier seed (Concealed from the Signer)
$m_{\text{Addr}}, H_{\text{Addr}}$	Except Signer	User's Address (Concealed from the Signer)

4.5 Protocol Workflow

4.5.1 Phase 1: Initialization (Init)

- **Issuer** generates $\ell + 3$ independent random generators:

$$H_0, H_1, \ldots, H_{\ell-1}, H_\gamma, H_{\text{null}}, H_{\text{Addr}} \xleftarrow{\$} \mathbb{G}_1$$

and a mapping between the explicit identity constraints and the corresponding message generators ($\vec{m}$), for example, constraint *Age>18* could be mapped to keccak256("$Age > 18$"), then publishes them as public parameters.

- **Signer** executes the KeyGen algorithm to sample a secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and publishes the corresponding public key $X = x \cdot G_2$.
- **Signer** or designated **Issuer** deploys the *Verifier* contract, embedding the public parameters $\{H_i\}, H_\gamma, H_{\text{null}}, H_{\text{Addr}}$, and public key X.

Current Status

- **Public Parameters**: $H_0, \ldots, H_{\ell-1}, H_\gamma, H_{\text{null}}, H_{\text{Addr}}, X, \vec{m}$
- **Signer Status**: Secret key x held locally.

4.5.2 Phase 2: User Application (Apply)

- **User**:
  1. Retrieves mandatory parameters and identity constraints ($\vec{m}$) using `view` calls against the target contract.
  2. Samples random scalars $m_{\text{null}}, m_\gamma \xleftarrow{\$} \mathbb{Z}_p$, and derives the scalar parameter m_{Addr} from $0x$UserAddr.
  3. Computes a foundational commitment covering the identity variables:
$$\mathcal{C}_2 = m_{\text{Addr}} \cdot H_{\text{Addr}} + m_{\text{null}} \cdot H_{\text{null}} + m_\gamma \cdot H_\gamma$$
  4. Aggregates the plaintext evidence vector $\vec{m} = \{m_0, \ldots, m_{\ell-1}\}$ and transmits $(\vec{m}, \mathcal{C}_2)$ **off-chain** to the **Signer**.

Current Status

- **Public Parameters**: $H_{\text{null}}, X, \vec{m}$
  - **Verifier Contract Context**:
- **Signer Scope**: $x, \langle \text{UserID}, \vec{m}, \mathcal{C}_2 \rangle$ pending review.
- **User Scope**: $0x$UserAddr, $m_{\text{null}}, m_\gamma$ generated.

4.5.3 Phase 3: Partial Blind Signature Issuance (Sign)

- **Signer**:
  1. Verifies that the explicit constraints ($\vec{m}$) correspond to the registered attributes of the UserID. If any check fails, the process is aborted.
  2. Computes the public commitment segment for the known attributes:
$$\mathcal{C}_1 = G_1 + \sum_{i=0}^{\ell-1} m_i \cdot H_i \in \mathbb{G}_1$$
  3. Aggregate the blinded commitment for signing:
$$\mathcal{C} = \mathcal{C}_1 + \mathcal{C}_2$$

4. Samples $e \xleftarrow{\$} \mathbb{Z}_p$ and computes the final signature components:

$$A = \frac{\mathcal{C}}{x+e} = \frac{\mathcal{C}_1 + \mathcal{C}_2}{x+e}$$

5. Returns the final signature $\sigma = (A, e)$ to the User to conclude the issuance phase.

Current Status

- **Public Parameters**: $H_{\text{null}}, X, \vec{m}$
- **Signer Scope**: $x, \langle \text{UserID}, \mathcal{C}_2, \sigma \rangle$
- **User Scope**: $0x\text{UserAddr}, m_{\text{null}}, m_\gamma, \sigma$

4.5.4 Phase 4: Proof Generation (NIZK Proof)

- **User**:

1. Constructs bounds and proofs to formulate a verifiable **NIZK Proof** $\pi = \left(\overline{A},\ \overline{B},\ U,\ s,\ t,\ u_\gamma\right)$ demonstrating possession of the signature over $(\vec{m}, m_{\text{null}}, m_\gamma)$ while preserving zero-knowledge for (m_γ, A, e).

   Using index arrays for tracking explicit readable vectors $J = \{0, 1, \ldots, \ell - 1, \text{null}, \text{Addr}\}$ versus hidden identifiers $I = \{\gamma\}$:

$$\begin{aligned} \mathcal{C} &= \mathcal{C}_J + \mathcal{C}_I \\ &= G_1 + \sum_{j \in J} m_j \cdot H_j + m_\gamma \cdot H_\gamma \end{aligned}$$

   Executes the Fiat-Shamir transformation sequence:

   - $\overline{A} = r \cdot A$ (Randomized signature point representation)
   - $\overline{B} = r \cdot \mathcal{C} - r \cdot e \cdot A$
   - $U = \alpha \cdot \mathcal{C}_J + \beta \cdot \overline{A} + \delta_\gamma \cdot H_\gamma$
   - $c = H(\mathsf{ctx} \parallel \vec{m}_J \parallel \overline{A} \parallel \overline{B} \parallel U)$
   - $s = \alpha + r \cdot c, \quad t = \beta - e \cdot c, \quad u_\gamma = \delta_\gamma + r \cdot m_\gamma \cdot c$

2. Formats the generated proof payload $(\pi, m_{\text{null}}, 0x\text{UserAddr})$ and submits it to the **Verifier** contract.

Current Status

- **Public Parameter Component**: $H_{\text{null}}, X, \vec{m}$
  - **Verifier Contract Context**: $\langle \pi, m_{\text{null}}, 0x\text{UserAddr} \rangle$.
- **Signer Scope**: $x, \langle \text{UserID}, \mathcal{C}_2, \sigma \rangle$
- **User Scope**: $0x\text{UserAddr}, m_{\text{null}}, m_\gamma, \sigma, \pi$

4.5.5 Phase 5: On-chain Verification and Minting (**Verify**)

- **Verifier**:

  1. Reconstructs the public commitment segment from the parameters $J = \{0, 1, \ldots, \ell - 1, \text{null}, \text{Addr}\}$ revealed on-chain:

$$\mathcal{C}_J = G_1 + \sum_{i=0}^{\ell-1} m_i \cdot H_i + m_{\text{null}} \cdot H_{\text{null}} + m_{\text{Addr}} \cdot H_{\text{Addr}}$$

  2. **Nullifier Security Assertions**:
     - Computes the nullifier hash $\mathcal{N} = \text{Hash}(0x\text{UserAddr} \parallel m_{\text{null}})$.
     - Validates its unspent status (`Used[N] == false`) to prevent double-spending.
  3. **Calculates NIZK Confirmations:**
     (a) Recalculates the challenge scalar from the Fiat-Shamir context:

$$c' = H(\mathsf{ctx} \parallel \vec{m}_J \parallel \overline{A} \parallel \overline{B} \parallel U)$$

     (b) **Pairing Equations** to check the signature binding relations:

$$e(\overline{A}, X) \stackrel{?}{=} e(\overline{B}, G_2)$$

     (c) **Homomorphic Structural Integrity** verification for algebraic consistency:

$$U + c' \cdot \overline{B} \stackrel{?}{=} s \cdot \mathcal{C}_J + t \cdot \overline{A} + u_\gamma \cdot H_\gamma$$

     Accept the proof if and only if these equivalences hold true.
  4. **Smart Contract Execution**:
     - Updates the state registry by recording `Used[N] = true`.
     - Triggers the creation function `mintSBT(0xUserAddr, attributes)` to assign the verifiable credential.

Current Status

- **Public Parameter Component**: $H_{\text{null}}, X, \vec{m}$
  - **Verifier Contract Context**: $\langle \pi, m_{\text{null}}, 0x\text{UserAddr}, \rangle$
- **Signer Scope**: $x, \langle \text{UserID}, \mathcal{C}_2, \sigma \rangle$
- **User Scope**: $0x\text{UserAddr}, m_{\text{null}}, m_\gamma, \mathcal{N}, \sigma, \pi, \text{TokenID}$

5 Security Analysis

In this section, we analyze the security properties of our proposed scheme and address potential vulnerabilities, ensuring unforgeability, unlinkability, and resistance against blockchain-specific attacks.

We evaluate our scheme within the standard provable security framework. Let $\lambda \in \mathbb{N}$ denote the security parameter, which concretely corresponds to the bit-length of the prime order p of the underlying cryptographic groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ (i.e., $|p| \approx \lambda$). It

fundamentally dictates the size of the key space and the mathematical hardness of the underlying algebraic problems (e.g., Discrete Logarithm, q-SDH).

A function $\text{negl}(\lambda)$ is termed *negligible* if, for every positive polynomial $\text{poly}(\cdot)$, there exists an integer N_c such that for all $\lambda > N_c$, it holds that $|\text{negl}(\lambda)| < \frac{1}{\text{poly}(\lambda)}$. Because the best-known algorithms to solve the underlying hard problems on pairing-friendly curves (such as Pollard's rho) require exponential time bounded by $\mathcal{O}(2^{\lambda/2})$, the success probability (Advantage, Adv) of any Probabilistic Polynomial-Time (PPT) adversary $\mathcal{A}$ evaluating within polynomial time is inherently asymptotically bounded by $\text{negl}(\lambda)$. A scheme is considered formally secure if $\text{Adv}(\mathcal{A}) \leq \text{negl}(\lambda)$.

5.1 Existential Unforgeability against Chosen Message Attacks (EUF-CMA)

Unforgeability ensures that no adversary, including unauthorized users or colluding external attackers, can compute a valid credential proof without the explicit authorization of the *Signer*.

Definition 1 (System Unforgeability Game).

Game $\mathsf{Sys\text{-}EUF\text{-}CMA}_{\mathcal{A}}(\lambda)$:

1. Setup: The challenger $\mathcal{C}$ runs $\mathsf{Init}(1^\lambda)$ to generate the parameter set $\{\{H_i\}, H_\gamma, H_{\text{null}}, H_{\text{Addr}}\}$ and keypair (x, X). It provides $\mathcal{A}$ with X and the public parameters.
2. Queries: $\mathcal{A}$ is given access to an interactive blind issuance oracle $\mathcal{O}_{\text{Sign}}$. $\mathcal{A}$ submits blinding commitments $\mathcal{C}_2^{(i)}$ and explicit attributes $\vec{m}_{\text{exp}}^{(i)} = (m_0^{(i)}, \ldots, m_{\ell-1}^{(i)})$. $\mathcal{O}_{\text{Sign}}$ computes the corresponding partial blind signatures σ_i. Let $\mathcal{Q}$ denote the set of successfully signed explicit attribute vectors.
3. Forgery: $\mathcal{A}$ outputs an on-chain verification payload $(\pi^*, \vec{m}_{\text{exp}}^*, m_{\text{null}}^*, m_{\text{Addr}}^*, M^*)$.
4. Verify: $\mathcal{A}$ **wins** if $\mathsf{Verify}(\mathsf{pp}, X, \pi^*, M^*, m_{\text{null}}^*, m_{\text{Addr}}^*) = 1$ and $\vec{m}_{\text{exp}}^* \notin \mathcal{Q}$.

In this game, the adversary attempts to produce a valid NIZK proof π^* for an explicit attribute vector $\vec{m}_{\text{exp}}^*$ that was never authorized by the Signer, simulating a user trying to forge an identity claim.

Theorem 1. *Our proposed CUBE scheme is Sys-EUF-CMA secure under the q-Strong Diffie-Hellman (q-SDH) assumption and the Simulation-Extractability of the NIZK proof system. For any PPT adversary $\mathcal{A}$, its advantage is bounded by:*

$$\text{Adv}_{\mathcal{A}}^{\text{Sys-EUF-CMA}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{q\text{-SDH}}(\lambda) + \text{Adv}_{\mathcal{E}}^{\text{Extract}}(\lambda) + \text{negl}(\lambda)$$

Proof. We prove this by constructing a reduction algorithm $\mathcal{B}$ that uses a successful adversary $\mathcal{A}$ to solve the q-SDH problem.

Setup: $\mathcal{B}$ receives a q-SDH instance. Following the proof framework in [TZ23], $\mathcal{B}$ embeds the q-SDH challenge into the public key X and generators $\{H_i\}, H_\gamma, H_{\text{null}}, H_{\text{Addr}}$, and provides these to $\mathcal{A}$.

Oracle Simulation: When $\mathcal{A}$ queries $\mathcal{O}_{\text{Sign}}$ with $(\mathcal{C}_2, \vec{m}_{\text{exp}})$, $\mathcal{B}$ does not know the secret key x nor the hidden attributes inside $\mathcal{C}_2$. However, by utilizing the algebraic structure of the simulated generators, $\mathcal{B}$ can answer signing queries for up to $q-1$ adaptively chosen vectors without knowing x, successfully simulating the interactive partial blind signature phase.

Forgery Extraction: Eventually, $\mathcal{A}$ outputs a forgery $(\pi^*, \vec{m}_{\text{exp}}^*, m_{\text{null}}^*, m_{\text{Addr}}^*)$. Because the NIZK proof system is Simulation-Extractable (or satisfies Proof of Knowledge),

there exists an efficient knowledge extractor $\mathcal{E}$ that, by rewinding $\mathcal{A}$ (via the Forking Lemma in the Random Oracle Model), can extract the underlying witness: the valid partial blind signature $\sigma^* = (A^*, e^*)$ and the ephemeral randomizer m_γ^*.

Solving q-SDH: Since $\vec{m}_{\text{exp}}^* \notin \mathcal{Q}$, the extracted signature σ^* is a valid BBS signature on a novel, unqueried attribute combination (including the explicit attributes and the extracted hidden elements). $\mathcal{B}$ can algebraically transform σ^* to extract the solution to the original q-SDH challenge. Therefore, any successful forgery by $\mathcal{A}$ directly constructs a solution for the q-SDH problem, with the overall bounds adhering to the extractor's negligible failure probability. □

5.2 Signer-Blindness and On-Chain Unlinkability

Blindness ensures that the Signer does not learn the blinded attributes $(m_{\text{Addr}}, m_{\text{null}})$ during the interactive issuance phase. **Unlinkability** ensures that even a fully compromised Signer, possessing a comprehensive transcript of all off-chain issuing requests, cannot link a newly submitted on-chain NIZK proof back to a specific issuance session.

Our protocol achieves perfect blindness intrinsically through the Pedersen commitment $\mathcal{C}_2$. We formally model the Unlinkability below:

Definition 2 (Unlinkability Game).

Game $\mathsf{Unlink}_{\mathcal{A}}(\lambda)$:

1. Setup: The challenger $\mathcal{C}$ runs $\mathsf{KeyGen}(1^\lambda)$ to obtain (x, X), and publishes X together with all system parameters to the adversary $\mathcal{A}$ (who acts as a malicious Signer).
2. User Registration: Two honest users U_0 and U_1 are initialized. Each U_b samples independent hidden parameters $(m_{\text{Addr}}^{(b)}, m_{\text{null}}^{(b)}, m_\gamma^{(b)})$ and submits a commitment $\mathcal{C}_{2,b} = m_{\text{Addr}}^{(b)} \cdot H_{\text{Addr}} + m_{\text{null}}^{(b)} \cdot H_{\text{null}} + m_\gamma^{(b)} \cdot H_\gamma$ together with identical explicit attributes $\vec{m}$ to $\mathcal{A}$.
3. Signature Issuance: $\mathcal{A}$ (acting as Signer) returns valid signatures $\sigma_0 = (A_0, e_0)$ and $\sigma_1 = (A_1, e_1)$ to U_0 and U_1 respectively.
4. Challenge: $\mathcal{C}$ flips a fair coin $b \xleftarrow{\$} \{0, 1\}$ and generates an on-chain NIZK proof π_b using the credentials of U_b. $\mathcal{C}$ sends π_b and the revealed public segment $\vec{m}_J$ to $\mathcal{A}$.
5. Guess: $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.
6. $\mathcal{A}$ **wins** if $b' = b$.

The Unlinkability Game models the ability of a fully malicious Signer—who possesses the complete off-chain issuance transcript including both $\mathcal{C}_{2,0}$ and $\mathcal{C}_{2,1}$, as well as the corresponding signatures—to trace an anonymous on-chain proof π_b back to the identity of the user who requested it. A secure scheme ensures that the Signer's advantage over random guessing remains negligible.

Theorem 2. *The proposed commitment-hiding issuance mechanism (with ephemeral randomizer m_γ) achieves computational signer-blindness and on-chain unlinkability. The advantage of $\mathcal{A}$ is bounded by:*

$$\text{Adv}_{\mathcal{A}}^{\text{Blind}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

Proof. The proof is twofold, addressing the unlinkability of both the off-chain issuance process and the on-chain verification proof.

Step 1: Perfect Hiding of the Off-Chain Commitment. The core of our off-chain issuance relies on the commitment:

$$\mathcal{C}_2 = m_{\text{Addr}} \cdot H_{\text{Addr}} + m_{\text{null}} \cdot H_{\text{null}} + m_\gamma \cdot H_\gamma.$$

This structure is a generalized Pedersen commitment [Ped92] over $\mathbb{G}_1$. For any given values of the identity parameters $(m_{\text{Addr}}, m_{\text{null}})$, and for any observed commitment $\mathcal{C}_2$, there uniquely exists a scalar m_γ mapping the attributes to $\mathcal{C}_2$. Because the ephemeral randomizer $m_\gamma \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen uniformly at random by the User and kept hidden from the Signer, the distribution of $\mathcal{C}_2$ is computationally independent of $(m_{\text{Addr}}, m_{\text{null}})$. Thus, the commitment perfectly hides the user's explicit identifiers from the Signer. Consequently, the conditional probability that a transcript matches U_0 or U_1 given $\mathcal{C}_{2,b}$ is exactly identical.

Step 2: Zero-Knowledge Property of the On-Chain NIZK. During the Verify phase, the user submits an NIZK proof π_b. The proof incorporates randomization elements, such as computing $\overline{A} = r \cdot A$ with $r \xleftarrow{\$} \mathbb{Z}_p^*$. Given the Perfect Honest-Verifier Zero-Knowledge (HVZK) property of the underlying proof protocol (validated in [TZ23]), a simulator lacking any witness can generate a proof distribution that is perfectly indistinguishable from genuinely generated proofs π_b. Therefore, the on-chain published elements π_b leak mathematically zero information regarding the underlying commitment scalars or the specific signature σ_b.

By compounding the perfect hiding property of $\mathcal{C}_2$ and the zero-knowledge nature of π_b, the entire view accessible to the malicious Signer $\mathcal{A}$ provides zero probabilistic advantage in distinguishing $b = 0$ from $b = 1$. Therefore, we conclude unconditionally that:

$$\text{Adv}_{\mathcal{A}}^{\text{Blind}}(\lambda) = 0 \leq \text{negl}(\lambda). \quad \square$$

5.3 NIZK Relation and Computational Soundness

To formally establish the soundness of our NIZK proof system, we first define the corresponding NP relation $\mathcal{R}_{\text{CUBE}}$. The public statement evaluated by the on-chain smart contract is $\mathsf{stmt} = (X, \{H_i\}, \mathcal{N}, \mathsf{addr}, \vec{m}_{\text{exp}}, M)$, and the private witness held by the user is $\mathsf{wit} = (A, e, m_{\text{null}}, m_\gamma)$. The relation $\mathcal{R}_{\text{CUBE}}(\mathsf{stmt}, \mathsf{wit}) = 1$ if and only if:

$$e(A, X + eG_2) = e(G_1 + \mathsf{addr} \cdot H_{\text{Addr}} + m_{\text{null}} \cdot H_{\text{null}} + m_\gamma \cdot H_\gamma + \sum m_i H_i, G_2)$$
$$\mathcal{N} = \text{Hash}(\mathsf{addr} \parallel m_{\text{null}})$$

This property guarantees that a malicious user cannot fabricate a valid proof π without knowing the actual underlying witnesses $(A, e, m_{\text{null}}, m_\gamma)$.

Theorem 3. *The NIZK scheme employed in CUBE provides Perfect Honest-Verifier Zero-Knowledge (Perfect HVZK) and Simulation-Extractability in the Random Oracle Model (ROM).*

Proof. The proof protocol implements a Fiat-Shamir transformed Σ-protocol derived from the ZKPoK framework in [TZ23]. For simulation-extractability (soundness in the ROM), suppose a computationally bounded adversary $\mathcal{A}$ can produce a valid proof $\pi = (\overline{A}, \overline{B}, c, s_e, s_{r2}, s_{r3}, \dots)$ for a false statement with non-negligible advantage.

By applying the General Forking Lemma [FS87], treating the cryptographic hash function as a Random Oracle, we construct a Knowledge Extractor $\mathcal{E}$. $\mathcal{E}$ observes $\mathcal{A}$'s interactions with the Random Oracle. Whenever $\mathcal{A}$ requests the challenge hash for a commitment prefix uniformly, $\mathcal{E}$ records the state. $\mathcal{E}$ then rewinds $\mathcal{A}$ to the exact state immediately before the challenge query, and injects a different random challenge $c' \neq c$. If $\mathcal{A}$ succeeds again, $\mathcal{E}$ obtains a second valid transcript for the identical initial commitments $(\overline{A}, \overline{B})$ but with distinct challenge and responses $(c', s'_e, s'_{r2}, s'_{r3}, \dots)$.

Given the linear structure of the verification equations, $\mathcal{E}$ algebraically subtracts the two transcript equations. For the scalar e in the BBS signature, $\mathcal{E}$ extracts it by computing $e = \frac{s_e - s'_e}{c - c'} \bmod p$. Identical linear extraction reconstructs the exact $m_{\text{null}}, m_\gamma$ and the randomized signature component A. If the extracted values do not satisfy $\mathcal{R}_{\text{CUBE}}$, it directly implies a collision in the Random Oracle or a break on the underlying group's generalized Discrete Logarithm problem. Consequently, the valid generation of π rigidly assures knowledge of the witness, yielding $\text{Adv}_{\mathcal{A}}^{\text{Soundness}}(\lambda) \leq \text{negl}(\lambda)$. □

5.4 Non-Transferability and Sybil Resistance

Blockchain applications mandate strict defense mechanisms against token commodification and Sybil identities.

Theorem 4. *The CUBE formulation guarantees non-transferability and double-spending (Sybil) resistance, supported by the Special Soundness of the NIZK proof system and the Collision-Resistance of the underlying hash function.*

Proof. A successful Sybil or transferability attack requires an adversary to either: (1) forge a valid proof for an unverified address $m'_{\text{Addr}} \neq m_{\text{Addr}}$; or (2) generate a second valid token by reusing the same credential but computing a different nullifier $\mathcal{N}' \neq \mathcal{N}$ to bypass the on-chain registry.

For the first case, as established in **Theorem 3**, the NIZK proof guarantees Special Soundness. The proof inherently verifies that the hidden attributes m_{Addr} and m_{null} match the committed signature components. Any attempt to verify under a modified address requires the adversary to forge the relation without knowing a valid witness, which breaks the Discrete Logarithm assumption and the BBS unforgeability, yielding $\text{Adv}_{\mathcal{A}}^{\text{Transfer}}(\lambda) \leq \text{negl}(\lambda)$.

For the second case, the contract mandates and irreversibly logs the generated deterministic nullifier $\mathcal{N} = \text{Hash}(0x\text{UserAddr} \parallel m_{\text{null}})$. Modifying m_{null} would invalidate the NIZK proof (due to soundness), whereas fabricating a collision $\text{Hash}(0x\text{UserAddr} \parallel m_{\text{null}}) = \text{Hash}(0x\text{UserAddr}' \parallel m'_{\text{null}})$ directly violates the collision-resistance of the hash function, constraining $\text{Adv}_{\mathcal{A}}^{\text{Sybil}}(\lambda) \leq \text{negl}(\lambda)$. Consequently, the strict coupling of NIZK soundness and deterministic nullifiers guarantees resistance against Sybil identity inflation. □

5.5 Resistance to Practical Threat Models

Beyond standard cryptographic unforgeability, deploying decentralized identities on public blockchains introduces realistic operational threats. We explicitly address them as follows:

- **Signer and User Collusion:** A malicious or bribed Signer can trivially issue a valid partial blind signature to an arbitrarily chosen user, bypassing off-chain checks (e.g., KYC). However, because the underlying signature utilizes unique keys, this collusion cannot compromise or frame other independent honest users (*Non-framability*). Further mitigations, such as distributing the issuance key across a committee via Threshold BBS+ [DKL+23], can structurally prevent single-node corruption.
- **Selective Issuance (Censorship):** A malicious Signer could arbitrarily refuse to produce signatures or drop requests. Our blind commitment ($\mathcal{C}_2$) mitigates targeted censorship because the Signer cannot see the specific address or nullifier during the Sign interaction. Systemic censorship (e.g., denial of service) must be handled by public SLA auditability or threshold architectures.
- **Malicious Verifier Contract Upgrades:** A compromised on-chain Verifier contract could bypass signature checks to forcibly mint invalid SBTs. To prevent this, our threat model assumes the Verifier and Issuer contracts are immutable, or their upgrade

pathways are governed by a robust DAO-controlled Timelock proxy. This ensures that any malicious logic replacement triggers a sufficient challenge period, granting users time to audit the new code or exit the platform.

6 Performance Evaluation

In this section, we evaluate the performance of the proposed scheme, focusing on off-chain computational overhead and on-chain verification costs, using a Rust-based WebAssembly (WASM) implementation and Solidity smart contracts.

6.1 Experimental Setup

Benchmarks were conducted on a consumer-grade machine (Apple M5 CPU, 16GB RAM, macOS Tahoe v26.4.1).

The off-chain cryptographic primitives are implemented in Rust (v1.95.0) and compiled into WebAssembly (WASM) via `wasm-bindgen` (v0.14.0) for execution within browser environments. The on-chain verification smart contracts are authored in Solidity (v0.8.33) and profiled using the Foundry framework, with the compiler optimizer enabled (200 runs).

Our implementation operates over the BN254 pairing-friendly elliptic curve to ensure EVM compatibility. This allows on-chain verification to directly leverage Ethereum's native precompiled contracts (`0x06`, `0x07`, and `0x08`) for elliptic curve point addition, scalar multiplication, and optimal Ate pairing checks. The complete source code is open-sourced and publicly available at https://github.com/letsverify.

6.2 Off-Chain Computational Overhead

We benchmark the latency of the off-chain cryptographic primitives in two environments: a native Rust runtime (representing backend entities like Issuers and Verifier nodes) and a WebAssembly (WASM) engine (representing client applications like web browsers). Benchmarks are conducted over varying numbers of attributes, specifically $|M| \in \{5, 10, 20\}$.

Figure 3 presents the timing metrics for the key cryptographic phases: key pair generation, interactive signature issuance (user blind commitment setup and issuer partial signing), non-interactive zero-knowledge proof (PoK) generation, and proof verification.

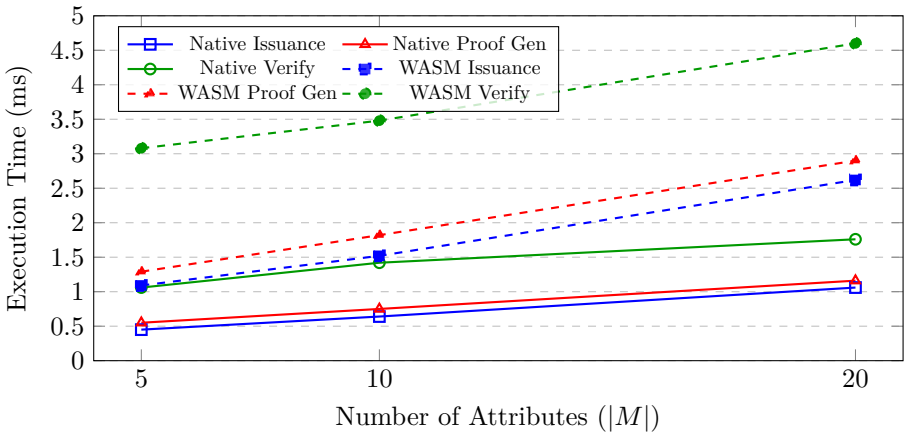


Figure 3: Off-Chain Execution Time of Cryptographic Phases across Attributes

The empirical results in Figure 3 indicate that algorithm performance scales linearly with the number of attributes $|M|$, except for the Key Generation phase which requires

constant $O(1)$ time. The WASM execution incurs a 2.5× to 3× performance degradation compared to the native implementation due to browser sandbox overhead. Nonetheless, the maximum observed latency for proofs with 20 attributes remains under 5 milliseconds. This low processing overhead demonstrates the practical feasibility of the scheme for decentralized identity applications.

6.3 On-Chain Verification Cost

We analyze the gas consumption of the Solidity smart contracts deployed on the Sepolia testnet. The on-chain verification relies on the EVM BN254 precompiled contracts to evaluate the zero-knowledge proofs.

6.3.1 Contract Deployment Costs

The system logic is decoupled into two primary contracts: 1. **BBSVerifier:** A stateless contract containing the cryptographic pairing logic required to validate the NIZK proofs. 2. **IssuerVerifier:** The business logic contract responsible for enforcing credential rules, verifying nullifier uniqueness, and minting the resulting Soulbound Tokens (ERC-721/ERC-5484).

Deploying the `BBSVerifier` consumes approximately 1.63 million gas, while the `IssuerVerifier` requires approximately 3.20 million gas. The total deployment overhead is roughly 4.83 million gas. Because deployment is a one-time procedure typically executed by credential issuers, this cost is a reasonable fixed expense.

6.3.2 Proof Verification Gas Cost

On-chain proof verification algorithm requires validating a bilinear pairing equation and checking homomorphic elliptic curve consistency. Table 1 details the gas consumption required to verify proofs with varying numbers of attributes, detailing the exact count of EVM precompiled contract invocations.

Table 1: On-Chain Proof Verification Gas Cost and Operation Breakdown

Attributes ($\|M\|$)	Total Gas	Est. Cost (USD)∗	`ecPairing`	`ecMul`	`ecAdd`
5	256,007	\$0.0619	1	9	8
10	319,157	\$0.0796	1	14	13
20	445,948	\$0.1076	1	24	23

∗Estimated fiat costs calculated based on prevailing Ethereum network rates as of May 6, 2026, 19:00 UTC+8.

We observe that the gas cost comprises a fixed overhead and a linearly scaling variable cost. The fixed overhead is dominated by the optimal Ate pairing check (`ecPairing` at `0x08`), consuming approximately 112,000 gas for two pairings, along with state hashing and memory allocations. The variable cost scales with the number of attributes ($|M|$): aggregating the commitment sequence requires one elliptic curve scalar multiplication (`ecMul` at `0x07`, $\sim 6,000$ gas) and one point addition (`ecAdd` at `0x06`, $\sim$150 gas) per attribute. Analytically, verifying a proof of $|M|$ attributes requires exactly 1 `ecPairing`, $|M| + 4$ `ecMul`, and $|M| + 3$ `ecAdd` operations.

Verifying a credential with 20 attributes incurs a total cost of $\sim$446k gas (approximately \$0.108 USD). This indicates that the proof verification remains well within Ethereum block limits and offers reasonable transaction fees for decentralized identity applications.

7 Conclusion

In this paper, we proposed CUBE, a verifiable credential framework designed to support privacy-preserving identity management within the Web3 ecosystem. By combining partially blind BBS signatures with Non-Interactive Zero-Knowledge (NIZK) proofs, our scheme addresses the practical conflict between public on-chain verification and user privacy. The introduction of a deterministic nullifier tied to Soulbound Tokens (SBTs) prevents Sybil attacks and guarantees non-transferability without compromising on-chain unlinkability. Our evaluations on EVM-compatible networks confirm the system's practicality: leveraging native cryptographic precompiles keeps on-chain gas costs low, even when verifying credentials with up to 20 attributes.

While full anonymity protects users, it also presents challenges regarding accountability. For future work, we plan to investigate conditional traceability to address the potential for malicious behavior. A promising approach involves integrating Time-Lock Puzzles (TLP) or delay encryption [RSW96, MT19], where a user obscures a unique identity secret and submits it to the issuer during credential issuance. Under normal conditions, the user remains anonymous. However, upon detecting severe protocol violations, a designated entity (e.g., the issuer or a DAO) can expend predefined computational resources to solve the puzzle. This unlocks the secret, enabling the tracing and revocation of the offending user. Such a mechanism provides a configurable layer of accountability, balancing necessary privacy features with overall system integrity.

References

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 56–73, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 41–55, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BSCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pages 459–474. IEEE, 2014.

[CDL16] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. Cryptology ePrint Archive, Paper 2016/663, 2016. URL: https://eprint.iacr.org/2016/663.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[DKL+23] Jack Doerner, Yashvanth Kondi, Eysa Lee, Abhi Shelat, and LaKyah Tyner. Threshold bbs+ signatures for distributed anonymous credential issuance. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 773–789, 2023. doi:10.1109/SP46215.2023.10179470.

[ENR+26] Liam Eagen, Hy Ngo, Vikas Rushi, Ying Tong, Moven Tsai, and Janabel Xia. OpenAC: Open design for transparent and lightweight anonymous

credentials. Cryptology ePrint Archive, Paper 2026/251, 2026. URL: https://eprint.iacr.org/2026/251.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[LCKO19] Jiwon Lee, Jaekyoung Choi, Jihye Kim, and Hyunok Oh. SAVER: SNARK-friendly, additively-homomorphic, and verifiable encryption and decryption with rerandomization. Cryptology ePrint Archive, Paper 2019/1270, 2019. URL: https://eprint.iacr.org/2019/1270.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 620–649, Cham, 2019. Springer International Publishing.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[PSS19] Alexey Pertsev, Roman Semenov, and Roman Stromberg. Tornado cash: Privacy on ethereum. Tornado Cash Whitepaper. https://tornadoeth.cash/r/whitepaper-tornado-cash, 2019.

[RS+22] Drummond Reed, Markus Sabadello, et al. Decentralized identifiers (DID) v1.0. W3C Recommendation, 2022. URL: https://www.w3.org/TR/did-core/.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report Technical memo MIT/LCS/TR-684, MIT Laboratory for Computer Science, 1996. (Revision 3/10/96).

[SAB+19] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. URL: https://www.ndss-symposium.org/ndss-paper/coconut-threshold-issuance-selective-disclosure-credentials-with-applications-to-distributed-ledgers/.

[SLL+22] Manu Sporny, Dave Longley, Chad Lemire, et al. Verifiable credentials data model v1.1. W3C Recommendation, 2022. URL: https://www.w3.org/TR/vc-data-model/.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. Cryptology ePrint Archive, Paper 2023/275, 2023. URL: https://eprint.iacr.org/2023/275.

[WOB22] E Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. Decentralized society: Finding web3's soul. *Available at SSRN 4105763*, 2022.

[ZLB24] Yong Zhou, Hong Lei, and Zijian Bao. Eisdspa: An efficient and secure blockchain-based donation scheme with privacy protection and auditability. *IEEE Open Journal of the Communications Society*, 5:7498–7510, 2024. doi:10.1109/OJCOMS.2024.3504403.