Toward Verifiable Privacy in Decentralized Identity: A Formal Framework for Minimal Disclosure and Unlinkability

Yu Zhang
Northwestern Polytechnical University; Beijing Pushtime Technology Co., Ltd.
zy168612@163.com

Zongbin Wang
Beijing Infosec Technologies Co., Ltd.
pony_wang@hotmail.com

Abstract

This paper presents a formal framework for decentralized identity (DID), which achieves both minimal disclosure and session unlinkability under public verifiability. We instantiate this framework as PrivDID. In PrivDID, a user can prove a predicate about a committed attribute via a single ring signature, thereby hiding in an anonymity set dynamically selected from the public ledger. PrivDID builds on Pedersen commitments and binary-range encodings, and is proven secure in the random oracle model. It is fully W3C-compliant and requires no trusted setup. Implementation shows practical efficiency in storage, communication, and computation, confirming real-world feasibility.

1 Introduction

The evolution of decentralized identity (DID) systems [1–5] offers a compelling departure from traditional identity management models, primarily by mitigating the risks associated with centralized governance and single points of failure [6, 7]. By building upon blockchain and distributed ledger foundations [8–13], DID architectures return control of digital identity to users themselves, reducing reliance on trusted intermediaries [14, 15]. Ongoing standardization work, most notably the W3C DID specification [16], has laid important groundwork for cross-system compatibility and adoption. However, this shift toward decentralization also brings to light complex privacy implications that current system designs and standards are not formally equipped to handle.

While the W3C Verifiable Credentials guidelines rightly discourage placing sensitive user data directly on-chain, the alternative disclosure mechanisms they propose—such as omitting entire claims via Merkle proofs or issuing derived credentials—rely largely on syntactic constructs rather than cryptographically sound privacy guarantees [16]. In practice, proposed solutions avoid plaintext storage through techniques like storing hashes [17] or explicit verification predicates (i.e., context-based credentials) [18] on-chain. However, these approaches fail to achieve minimal disclosure [19–23]. More fundamentally, a rigorous security model that adequately captures both minimal disclosure and session unlinkability in a publicly auditable, decentralized context is still lacking, making it difficult to assess whether systems truly protect user data against longitudinal tracking.

1.1 Related Work

Existing proposals make critical trade-offs in their attempt to balance verifiability and privacy. [17] proposes a scheme that stores hashes of (user, identifier) pairs on-chain, where each identifier is used to retrieve the corresponding attribute. This decouples attributes from users for passive ledger observation. However, to verify any predicate, the user must disclose the full attribute value to the verifier. [18] records the truth value of specific attribute predicates directly on the ledger. Although this enables verification without revealing the attribute value itself, the on-chain predicate statements directly disclose properties about the user, constituting another form of privacy leakage.

These concrete but limited schemes highlight the need for a rigorous formal foundation. A notable advance in this direction is [24], which establishes a formal model for minimal disclosure and provides a concrete cryptographic construction based on commitments and zero-knowledge proofs [25–28]. Its security guarantees ensure that the on-chain commitments perfectly hide the attribute values, and the subsequent predicate proofs reveal no information beyond the truth value of the predicate. However, it explicitly does not address session unlinkability. This limitation is significant in practice, where users interact repeatedly with services, and such linkability could enable longitudinal tracking despite individual sessions preserving minimal disclosure.

Therefore, the central research question addressed in this work is: How can we formally define and provably achieve both minimal disclosure and session unlinkability in decentralized identity systems, while maintaining public verifiability?

1.2 Contributions

This paper makes the following key contributions:

- **A Unified Formal Framework for Privacy-Preserving DID.** We introduce a unified formal framework for decentralized identity that simultaneously addresses *minimal disclosure* and *session unlinkability*. Our framework defines four core security properties—Completeness, Consistency, Attribute-Based Indistinguishability, and Session Unlinkability, providing a rigorous foundation for analyzing privacy in blockchain-based identity systems.
- **A Concrete Cryptographic Construction.** We present a practical cryptographic scheme that instantiates our framework. The construction uniquely integrates Pedersen commitments with binary prefix encoding and ring signatures, allowing users to prove attribute predicates while remaining anonymous within a dynamically formed set of on-chain commitments. The design does not require trusted setup and maintains full compatibility with W3C DID standards.
- **Rigorous Security Proofs.** We provide formal security proofs demonstrating that our construction satisfies all four security properties under standard cryptographic assumptions—namely the hardness of the elliptic curve discrete logarithm problem and the anonymity of the underlying ring signature scheme. The proofs establish that the scheme reveals nothing beyond predicate truth values and prevents cross-session linkability.
- **Experimental Validation and Performance Analysis.** We implement a prototype of our scheme and conduct a thorough performance evaluation. Experimental results show that the additional overhead for unlinkability scales linearly with the anonymity set size, with proof generation and verification times remaining within practical limits for real-world deployment.

2 Problem Statement and System Model

2.1 Problem Statement

The core challenge addressed in this work is the inherent conflict between *public verifiability* and *user privacy* in decentralized identity (DID) systems. Our objective is to propose a cryptographic framework that allows a user to cryptographically demonstrate possession of credentials satisfying certain properties to a service provider, while enforcing the principle of *minimal disclosure* and ensuring *session unlinkability*. This must be achieved under the following constraints and security assumptions:

- **Public Verifiability:** To ensure global trust and auditability, all critical identity assertions—such as DID documents and credential metadata—must be recorded on a transparent, consensus-based ledger. This foundational requirement eliminates reliance on centralized authorities and provides non-repudiable evidence of credential state, but simultaneously exposes all recorded data to potential privacy analysis.
- **Privacy Objectives:** The system must enforce two complementary privacy guarantees. First, it should support *minimal disclosure*: during service access, a verifier should learn only the truth value of a predicate θ (e.g., "age ≥ 18") about a user's attribute, while the exact attribute value itself remains confidential. Second, it must ensure *session unlinkability*: multiple presentations of credentials, whether to the same or different verifiers, should not be correlatable to a single identity holder, thereby preventing long-term behavioral tracking.
- **Adversarial Model:** Our security analysis adopts the standard malicious adversary model, in which all protocol participants may arbitrarily deviate from the prescribed protocol. The sole exception is a specific trust assumption regarding the Identity Provider (IdP), which is assumed to preserve the confidentiality of user attributes it learns during credential issuance. This model forms the baseline for the formal security definitions in Section 2.3. In Section 4.5, we further discuss how standard external security mechanisms (e.g., nonces for replay protection) can be integrated to defend against broader system-level attacks in practical deployments.

2.2 System Model

2.2.1 Entities and Workflow

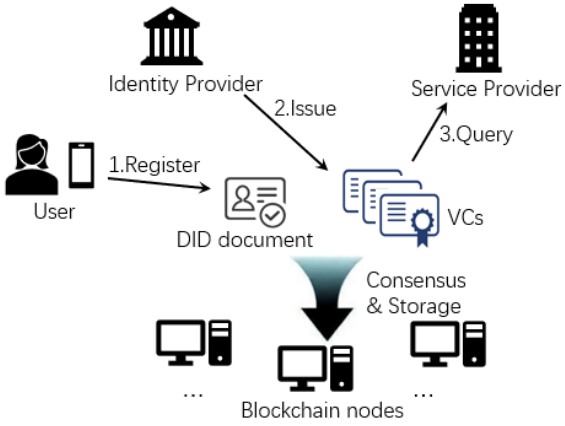


Figure 1: System architecture and operational workflow.

The system model, illustrated in Fig. 1, consists of three core entities whose interactions are coordinated through a public blockchain. The blockchain serves as a passive, immutable

ledger, providing the foundation for global verifiability and non-repudiation.

Core Entities:

- **User ($\mathcal{U}$):** The credential holder, who establishes a Decentralized Identifier (DID) and seeks to prove properties of their attributes.
- **Identity Provider (IdP):** The semi-trusted credential issuer, who attests to the validity of a user's attributes after off-chain verification. Our model assumes the IdP is honest in preserving attribute confidentiality, but does not require it to be trusted for other protocol operations.
- **Service Provider (SP):** The resource owner, who grants access based on verified proofs about user attributes.

The system operation follows three sequential phases.

1. **Registration and DID Anchoring.** The User ($\mathcal{U}$) creates a *DID Document* containing its public keys and registers it on the blockchain. This establishes $\mathcal{U}$'s globally resolvable cryptographic identity.
2. **Credential Issuance.** To obtain a credential for an attribute (e.g., age), $\mathcal{U}$ submits validated evidence to the IdP. After verification, the IdP issues a *Verifiable Credential (VC)*—a signed statement containing the commitment to the attribute value, not the value itself—and anchors it to the blockchain.
3. **Service Access and Verification.** To apply for service, $\mathcal{U}$ constructs an anonymity set from active commitments on chain and generates a zero-knowledge proof π_θ. This proof demonstrates, in zero-knowledge, both legitimate membership in the set and that $\mathcal{U}$'s committed attribute satisfies the service's policy θ. The SP first confirms via the blockchain that all commitments in set are unrevoked, then verifies the proof π_θ. Access is granted only on success; the SP learns solely that θ holds for the proof originator within the anonymity set, without identity or attribute disclosure.

The system also supports standard DID credential management operations, such as revocation and update, to ensure long-term practicality and ecosystem compatibility. A full treatment of these mechanisms is beyond the scope of this work, as they are orthogonal to the core privacy-preserving presentation protocol and do not weaken its formal guarantees.

2.2.2 Cryptographic Interface

We adhere to the standard cryptographic practice for verifiable credentials. A credential is considered *valid* if it carries a valid digital signature from its issuing Identity Provider (IdP). The verification of this signature uses the IdP's public key, which is obtained from its own DID document on the blockchain. This foundational mechanism for establishing credential authenticity is orthogonal to our core privacy contributions. Therefore, we omit the explicit steps of signature generation and verification from the algorithm descriptions below, focusing solely on the novel components.

To realize the privacy-preserving workflow, we define a suite of cryptographic algorithms that build upon this standard foundation. We first define the algorithm Protect that creates a protected attribute. On input public parameters pp, an attribute value v, and auxiliary input aux, it outputs a cryptographic commitment $\mathcal{C}$ and an opening proof π_{open}:

$$\mathsf{Protect}(pp, v, aux) \to (\mathcal{C}, \pi_{\text{open}})$$

where $\mathcal{C}$ is a commitment to the attribute value v, and π_{open} is a zero-knowledge proof certifying that $\mathcal{C}$ is correctly formed from v using aux. A corresponding algorithm VerifyOpen allows verification of this proof. The scheme must provide *hiding*, *binding*, and *zero-knowledge* properties.

Building upon this model, the system's cryptographic workflow is defined by six algorithms that concretely realize the three operational phases of Section 2.2.1. The *Registration and DID Anchoring* phase is initiated by the CreateDIDD algorithm. The *Credential Issuance* phase is carried out by the interactive algorithms ApplyCred and GenCred. Finally, the *Service Access and Verification* phase is implemented by the interactive pair ApplyServ and VerifyServ. The Setup algorithm generates the global public parameters required by all phases.

System Algorithms:

- $\mathsf{Setup}(1^\lambda) \to pp$: Generates and publishes public parameters pp. Executed once.
- $\mathsf{CreateDIDD}(pp) \to (\mathsf{didd}, pk, sk)$: Executed by a user to create a key pair and a DID Document didd for registration.
- $\mathsf{ApplyCred}(pp, v, aux) \to (\mathcal{C}, \pi_{\text{open}})$: Executed by a user. It calls $\mathsf{Protect}(pp, v, aux)$ to obtain $\mathcal{C}$ and π_{open}, to be sent to the IdP.
- $\mathsf{GenCred}(pp, \mathsf{didd}, \mathtt{attr}, v, \mathcal{C}, \pi_{\text{open}}, \mathsf{Proof}_v) \to VC$: Executed by the IdP. It first verifies the user's proof of attribute ownership Proof_v for the claimed value v, and the opening proof π_{open} for the commitment $\mathcal{C}$. If both are valid, it creates and signs a Verifiable Credential VC that cryptographically binds the attribute attr to $\mathcal{C}$ for the user identified by didd.
- $\mathsf{ApplyServ}(pp, v, aux, \theta, \mathcal{L}) \to (\pi_\theta, I)$: Executed by a user. From the ledger $\mathcal{L}$, it selects a set of credentials that includes its own. It outputs a proof π_θ and a descriptor I that uniquely identifies the selected set within $\mathcal{L}$. The proof π_θ asserts the satisfaction of θ while providing unlinkability.

- $\mathsf{VerifyServ}(pp, \mathcal{L}, I, \theta, \pi_\theta) \to \{0,1\}$: Executed by the SP. It uses I to retrieve the exact set of credentials from $\mathcal{L}$, then verifies π_θ to determine whether to grant access. It outputs 1 if the verification succeeds, and 0 otherwise.

2.3 Definitions

We now formalize the security and privacy requirements for our framework. Our definitions are structured in two tiers and apply to any protocol implementing the abstract interface of Section 2.2.2. The first tier adopts the standard notions for committed attributes from [24]: *consistency* (binding) prevents a commitment from being opened to two different values, while *attribute-based indistinguishability* hides the committed value itself. Together, they form the foundation for secure attribute commitments. The second tier introduces a new, stronger notion of *session unlinkability*, which prevents the linkage of different protocol sessions to the same user—a privacy dimension essential for practical identity systems and not addressed by predicate-based indistinguishability alone.

2.3.1 Completeness

A protocol Π is considered complete if an honest user who correctly follows Π's specification will successfully achieve the intended outcome in both credential issuance and service access. Formally, we define two complementary completeness experiments, which model credential generation and service access respectively.

Credential Generation Completeness ($\mathsf{Exp}^{\text{VC-COMP}}_{\Pi,\mathcal{S}}(\lambda)$): This experiment guarantees that an honest user with valid external evidence for an attribute v can always obtain a corresponding verifiable credential.

Experiment $\mathsf{Exp}^{\text{VC-COMP}}_{\Pi,\mathcal{S}}(\lambda)$:

1. **Initialization:** The challenger $\mathscr{C}$ generates $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and sends it to $\mathcal{S}$. $\mathscr{C}$ is given a valid tuple $(attr, v, aux, \mathsf{Proof}_v)$ representing the honest user's attribute and evidence.
2. **Simulator Input:** The simulator $\mathcal{S}$ generates and submits a ledger state $\mathcal{L}$ (containing DID documents and VCs) to $\mathscr{C}$.
3. **User Execution:** $\mathscr{C}$ computes $(\mathcal{C}, \pi_{\text{open}}) \leftarrow \mathsf{ApplyCred}(pp, v, aux)$.
4. **Issuance Attempt:** $\mathscr{C}$ sends $(didd, attr, v, \mathcal{C}, \pi_{\text{open}}, \mathsf{Proof}_v)$ to the simulator $\mathcal{S}$, who is required to execute $\mathsf{GenCred}$ honestly. The experiment outputs 1 if $\mathcal{S}$ does not return a valid VC, and 0 otherwise.

The scheme Π satisfies **credential generation completeness** if for all PPT simulators $\mathcal{S}$,

$$\mathsf{Adv}^{\text{VC-COMP}}_{\Pi,\mathcal{S}}(\lambda) = \Pr\left[\mathsf{Exp}^{\text{VC-COMP}}_{\Pi,\mathcal{S}}(\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

Service Access Completeness ($\mathsf{Exp}^{\text{SERV-COMP}}_{\Pi,\mathcal{S}}(\lambda)$): This experiment guarantees that an honest user, whose committed attribute v satisfies a predicate θ and whose commitment is present on the ledger, can always generate a valid proof that leads to access being granted.

Experiment $\mathsf{Exp}^{\text{SERV-COMP}}_{\Pi,\mathcal{S}}(\lambda)$:

1. **Initialization:** The challenger $\mathscr{C}$ generates $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and sends it to $\mathcal{S}$. $\mathscr{C}$ is given a tuple $(attr, v, aux, \mathsf{Proof}_v, \theta)$ where v satisfies θ.
2. **Credential Setup:** $\mathscr{C}$ computes $(\mathcal{C}, \pi_{\text{open}}) \leftarrow \mathsf{ApplyCred}(pp, v, aux)$ and interacts with $\mathcal{S}$ (acting as the IdP) to obtain a valid verifiable credential VC for $(attr, v, \mathcal{C})$. The ledger $\mathcal{L}$ is updated to include VC.
3. **Proof Generation:** $\mathscr{C}$ computes $(\pi_\theta, I) \leftarrow \mathsf{ApplyServ}(pp, v, aux, \theta, \mathcal{L})$.
4. **Verification Attempt:** The experiment outputs 1 if $\mathsf{VerifyServ}(pp, \mathcal{L}, I, \theta, \pi_\theta) \neq 1$, and 0 otherwise.

The scheme Π satisfies **service access completeness** if for all PPT simulators $\mathcal{S}$,

$$\mathsf{Adv}^{\text{SERV-COMP}}_{\Pi,\mathcal{S}}(\lambda) = \Pr\left[\mathsf{Exp}^{\text{SERV-COMP}}_{\Pi,\mathcal{S}}(\lambda) = 1\right] \le \mathsf{negl}(\lambda).$$

2.3.2 Experimental Framework and Oracles

The following security properties are formulated as interactive games between a challenger $\mathscr{C}$ and an adversary $\mathcal{A}$. All games share a common experimental setup. The challenger $\mathscr{C}$ maintains:

- Public parameters pp generated via $\mathsf{Setup}(1^\lambda)$.
- A simulated, publicly readable ledger $\mathcal{L}$, which $\mathscr{C}$ updates according to protocol operations.
- A private local state Σ that stores secret data (e.g., secret keys sk, randomness aux) for entities created during the game, ensuring $\mathscr{C}$ can faithfully simulate honest party behavior in subsequent steps.

The adversary $\mathcal{A}$ has read access to the public ledger $\mathcal{L}$ at any time. Additionally, $\mathcal{A}$ can adaptively query the following oracles maintained by $\mathscr{C}$, which update $\mathcal{L}$ and Σ as needed:

- $O_{\mathsf{CreateDIDD}}$: On request, generates $(didd, pk, sk) \leftarrow \mathsf{CreateDIDD}(pp)$, stores $(didd, sk)$ in Σ, registers $didd$ on $\mathcal{L}$, and returns $didd$.
- $O_{\mathsf{ApplyCred}}$: On input (v, aux), returns $(\mathcal{C}, \pi_{\text{open}}) \leftarrow \mathsf{ApplyCred}(pp, v, aux)$. $\mathscr{C}$ stores $(v, aux, \mathcal{C})$ in Σ for future queries (e.g., by $O_{\mathsf{ApplyServ}}$).
- O_{GenCred}: On input $(didd, attr, v, \mathcal{C}, \pi_{\text{open}}, \mathsf{Proof}_v)$, verifies the proof and evidence. If both are valid, it generates a verifiable credential VC, updates $\mathcal{L}$ with VC, and returns VC; else return $\perp$.

- $O_{\mathsf{ApplyServ}}$: On input $(didd, \theta)$, verifies that $didd$ is registered and retrieves the corresponding $(v, aux, \mathcal{C})$ from Σ. If v satisfies θ, it computes $(\pi_\theta, I) \leftarrow \mathsf{ApplyServ}(pp, v, aux, \theta, \mathcal{L})$ and returns (π_θ, I); else return $\perp$.

- $O_{\mathsf{VerifyServ}}$: On input (θ, π_θ, I), returns $\mathsf{VerifyServ}(pp, \mathcal{L}, I, \theta, \pi_\theta)$.

2.3.3 Consistency

The consistency property ensures the fundamental soundness of the credential system: a commitment $\mathcal{C}$ can only be validly opened to a single attribute value v. It is defined by the following experiment.

Experiment $\mathsf{Exp}^{\mathsf{CONS}}_{\Pi,\mathcal{A}}(\lambda)$:

1. $\mathscr{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain pp, initializes an empty public ledger $\mathcal{L}$ and an empty private state Σ. It gives pp to $\mathcal{A}$.

2. $\mathcal{A}$ is granted adaptive query access to the oracles $O_{\mathsf{CreateDIDD}}, O_{\mathsf{ApplyCred}}, O_{\mathsf{GenCred}}, O_{\mathsf{ApplyServ}}, O_{\mathsf{VerifyServ}}$. $\mathcal{A}$ also has read access to $\mathcal{L}$ at all times.

3. Eventually, $\mathcal{A}$ outputs a tuple $(v, v', \mathcal{C}, \pi_{\text{open}}, \pi'_{\text{open}})$ where $v \neq v'$.

4. The experiment outputs 1 if both $\mathsf{VerifyOpen}(\mathcal{C}, v, \pi_{\text{open}}) = 1$ and $\mathsf{VerifyOpen}(\mathcal{C}, v', \pi'_{\text{open}}) = 1$ hold. Otherwise, it outputs 0.

The *consistency advantage* of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{CONS}}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{CONS}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]$. A scheme Π satisfies **consistency** if for every PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{CONS}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

2.3.4 Attribute-Based Indistinguishability

The Attribute-Based Indistinguishability (AIND) property ensures that the on-chain commitment $\mathcal{C}$ reveals no information about the underlying sensitive attribute value v. It is defined by the following experiment.

Experiment $\mathsf{Exp}^{\mathsf{AIND}}_{\Pi,\mathcal{A}}(\lambda)$:

1. $\mathscr{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain pp, initializes an empty public ledger $\mathcal{L}$ and an empty private state Σ, sends pp to $\mathcal{A}$.

2. $\mathcal{A}$ is granted adaptive query access to the oracles $O_{\mathsf{CreateDIDD}}, O_{\mathsf{ApplyCred}}, O_{\mathsf{GenCred}}, O_{\mathsf{ApplyServ}}, O_{\mathsf{VerifyServ}}$ and read access to $\mathcal{L}$. This is the **query phase 1**.

3. $\mathcal{A}$ submits a **challenge** (v_0, v_1) to $\mathscr{C}$, where $v_0 \neq v_1$.

4. $\mathscr{C}$ samples a random bit $b \xleftarrow{\$} \{0,1\}$, samples fresh randomness aux_b, and computes the challenge commitment $(\mathcal{C}_b, \pi_{\text{open}}) \leftarrow \mathsf{ApplyCred}(pp, v_b, aux_b)$. It returns the commitment $\mathcal{C}_b$ to $\mathcal{A}$ (but not π_{open} or aux_b).

5. $\mathcal{A}$ may continue to query the oracles as in phase 1, with the following natural **restriction**: it cannot query O_{GenCred} or $O_{\mathsf{ApplyServ}}$ on inputs that would require opening $\mathcal{C}_b$ to either v_0 or v_1 (e.g., using $\mathcal{C}_b$ as part of a credential issuance or proof generation). This is the **query phase 2**.

6. Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$.

7. The experiment outputs 1 if $b' = b$, and 0 otherwise.

The *indistinguishability advantage* of $\mathcal{A}$ is defined as:

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

A scheme Π satisfies **AIND** if for every PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{AIND}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

2.3.5 Session Unlinkability

The Session Unlinkability property ensures that an adversary cannot determine whether two presented proofs were generated from the same credential or from two different credentials that satisfy the same predicate. It is defined by the following experiment.

Experiment $\mathsf{Exp}^{\mathsf{SUNL}}_{\Pi,\mathcal{A}}(\lambda)$:

1. $\mathscr{C}$ runs $\mathsf{Setup}(1^\lambda)$ to obtain pp, initializes an empty public ledger $\mathcal{L}$ and an empty private state Σ. It gives pp to $\mathcal{A}$.

2. $\mathcal{A}$ is granted adaptive query access to all oracles and read access to $\mathcal{L}$. This is the **query phase 1**.

3. $\mathcal{A}$ submits a **challenge request** $(didd_1, didd_2, \theta)$ to $\mathscr{C}$. Here, $didd_1$ and $didd_2$ must correspond to two distinct, valid credentials VC_1 and VC_2 on $\mathcal{L}$ (which $\mathcal{A}$ has obtained through prior oracle queries). Both underlying attribute values v_1 (for VC_1) and v_2 (for VC_2) must satisfy the predicate θ.

4. $\mathscr{C}$ retrieves the necessary data (v_1, aux_1) and (v_2, aux_2) from Σ for VC_1 and VC_2, respectively.

5. $\mathscr{C}$ samples a random bit $b \xleftarrow{\$} \{0,1\}$.

   - If $b = 0$ (same credential): $\mathscr{C}$ generates both proofs from VC_1:

     $$(\pi_1, I_1) \leftarrow \mathsf{ApplyServ}(pp, v_1, aux_1, \theta, \mathcal{L}),$$
     $$(\pi_2, I_2) \leftarrow \mathsf{ApplyServ}(pp, v_1, aux_1, \theta, \mathcal{L}).$$

- If $b = 1$ (different credentials): $\mathscr{C}$ generates the first proof from VC_1 and the second from VC_2:

$$(\pi_1, I_1) \leftarrow \mathsf{ApplyServ}(pp, v_1, aux_1, \theta, \mathcal{L}),$$
$$(\pi_2, I_2) \leftarrow \mathsf{ApplyServ}(pp, v_2, aux_2, \theta, \mathcal{L}).$$

In both cases, the descriptors I_1 and I_2 must be constructed to include the indices of *both* VC_1 and VC_2 from $\mathcal{L}$.

6. $\mathscr{C}$ returns the challenge tuple (π_1, I_1, π_2, I_2) to $\mathcal{A}$.
7. $\mathcal{A}$ may continue to query the oracles adaptively. This is the **query phase 2**.
8. Finally, $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$.
9. The experiment outputs 1 if $b' = b$, and 0 otherwise.

The *unlinkability advantage* of $\mathcal{A}$ is defined as:

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

A scheme Π satisfies **unlinkability** if for every PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{SUNL}}_{\Pi,\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

3 From Model to Instantiation: A Concrete Cryptographic Scheme

3.1 Bridging the Abstract and Concrete Layers

This instantiation provides a specific implementation of the abstract algorithms and interfaces defined in Section 2.2.2, using well-established cryptographic primitives. The following establishes the correspondence between the abstract and concrete layers of PrivDID:

- The abstract Protect operation, which generates a protected attribute, is concretely implemented in two steps: (1) generating a **Pedersen commitment set** for all binary prefixes of the attribute value v to form $\mathcal{C}$, and (2) producing a **non-interactive Schnorr proof of knowledge** π_{open} that demonstrates correct knowledge of the openings for the entire Pedersen commitment set. The algorithm outputs the pair $(\mathcal{C}, \pi_{\text{open}})$ as defined.
- Consequently, the abstract VerifyOpen operation is realized by **verifying the aforementioned Schnorr proof** π_{open} against the commitment $\mathcal{C}$.
- For service access, the predicate proof π_θ in ApplyServ and VerifyServ is implemented through a prefix encoding and ring signature mechanism. The user first encodes the predicate θ (e.g., $v \geq v_l$) into a set of covering binary prefixes. To prove satisfaction of θ, the user then uses a ring signature to demonstrate knowledge of a secret key corresponding to one of a set of derived public keys, which are constructed from the user's own and others' prefix commitments fetched from the public ledger. This process simultaneously proves the predicate and achieves unlinkability.

3.2 Mathematical Background and Preliminaries

Our construction utilizes standard cryptographic primitives. We briefly recall their definitions and key properties here, establishing the notation used throughout this section. For the core cryptographic operations, all operations are performed within a cyclic group $\mathbb{G}$ of prime order q on an elliptic curve, where the discrete logarithm problem is assumed to be hard. Let G be a generator of $\mathbb{G}$. We denote scalar multiplication of a point P by $a \in \mathbb{Z}_q$ as $[a]P$.

3.2.1 Pedersen Commitment

Given a second independent generator $H \in \mathbb{G}$, where the discrete logarithm relationship between G and H is unknown, a Pedersen commitment [29, 30] to a value $x \in \mathbb{Z}_q$ with randomness $r \in \mathbb{Z}_q$ is computed as:

$$\mathsf{Com}(x; r) = [x]H + [r]G.$$

- **Perfect Hiding:** For any x', there exists r' such that $\mathsf{Com}(x; r) = \mathsf{Com}(x'; r')$. Thus, the commitment reveals no information about x.
- **Computational Binding:** Under the Discrete Logarithm (DL) assumption in $\mathbb{G}$, it is computationally infeasible to find $(x, r) \neq (x', r')$ such that $\mathsf{Com}(x; r) = \mathsf{Com}(x'; r')$.

This scheme forms the basis for hiding user attribute values in our construction.

3.2.2 Non-Interactive Schnorr Proof

The non-interactive Schnorr proof protocol [31, 32] allows a prover $\mathcal{P}$ to demonstrate knowledge of a discrete logarithm $s \in \mathbb{Z}_q$ for a public key $Y = [s]G$ to a verifier $\mathcal{V}$.

- $\pi \leftarrow \textbf{Prove}(s | Y, G : Y = [s]G)$: $\mathcal{P}$ selects $k \xleftarrow{\$} \mathbb{Z}_q$, computes $R = [k]G$, $c = \mathcal{H}(R, Y)$, and $z = k + c \cdot s \mod q$. The proof is $\pi = (R, z)$.
- $\{0,1\} \leftarrow \textbf{Verify}(Y, G, \pi : Y = [s]G)$: $\mathcal{V}$ recomputes $c = \mathcal{H}(R, Y)$ and accepts iff $[z]G = R + [c]Y$.

The non-interactive Schnorr proof used in our construction is *honest-verifier zero-knowledge* and provides *simulation-extractability* in the random oracle model, assuming the hardness of the DL problem in $\mathbb{G}$. This directly implies its *existential unforgeability*. We use it to prove knowledge of the randomness opening a commitment.

3.2.3 Ring Signatures

A ring signature scheme [33–35] allows a signer to anonymously sign a message msg on behalf of a set (ring) of n public keys $R_{PK} = \{PK_1, \ldots, PK_n\}$, where the signer holds the secret key sk_s for $PK_s \in R_{PK}$.

- **Sign** $(sk_s, R_{PK}, m) \to \sigma$**:** Outputs a signature σ.
- **Verify** $(R_{PK}, m, \sigma) \to \{0,1\}$**:** Verifies the signature against the ring R_{PK}.

A secure ring signature scheme must provide:

- **Anonymity:** For any $PK_i \in R_{PK}$, the probability that an adversary identifies the true signer s is at most $1/n + \mathsf{negl}(\lambda)$.
- **Unforgeability:** Without knowing any sk_i for $PK_i \in R_{PK}$, it is infeasible to produce a valid (R_{PK}, m, σ) under the DL assumption.

This primitive is the cornerstone for achieving *session unlinkability* in our service access protocol.

3.2.4 Best Range-Covering (BRC)

Our construction builds upon the *Best Range-Covering (BRC)* technique [36, 37], which provides an efficient way to represent a continuous numerical range using a minimal set of nodes from a perfect binary tree.

Consider a perfect binary tree of height m with 2^m leaves. The leaf nodes, from left to right, correspond to the numerical domain $\mathcal{D} = \{0, 1, \ldots, 2^m - 1\}$, where leaf τ_i represents the value i. An internal node $\tau_{i,j}$ covers the contiguous interval $[i, j]$ formed by its leftmost and rightmost descendant leaves.

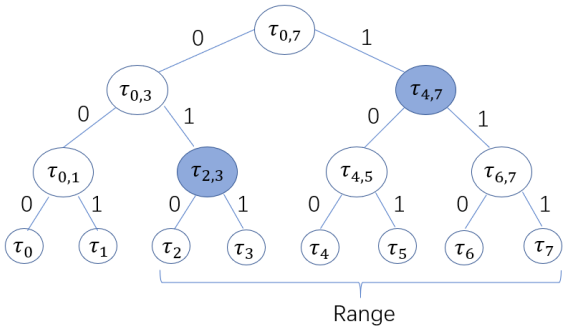


Figure 2: Example of BRC for range $[2,7]$ in a tree of height $m = 3$ ($\mathcal{D} = \{0, \ldots, 7\}$)

Given a range $[v_l, v_h] \subseteq \mathcal{D}$, its **Best Range-Covering (BRC)** is the *minimum set* of tree nodes whose union of covered intervals exactly equals $[v_l, v_h]$. This representation is compact and canonical. The BRC for a range can be computed using the standard $\mathsf{GetBRC}(v_l, v_h, m)$ algorithm [38,39], which outputs a set $\mathcal{B}$ of node identifiers (typically encoded as root-to-node paths).

The key property we leverage is the following: a value $v \in \mathcal{D}$ lies within $[v_l, v_h]$ if and only if the set of all binary prefixes of v's m-bit representation (denoted $\mathcal{P}_v$, obtained by $\mathsf{GetBinPrefixes}(v, m)$) contains **exactly one** element that also belongs to the BRC $\mathcal{B}$. Formally,

$$v \in [v_l, v_h] \quad \Longleftrightarrow \quad |\mathcal{P}_v \cap \mathcal{B}| = 1.$$

This is illustrated by the example in Fig. 2: for $[2,7]$, $\mathcal{B} = \{01, 1\}$, and for $v = 4$ (binary '100'), $\mathcal{P}_v = \{1, 10, 100\}$. The intersection $\mathcal{P}_v \cap \mathcal{B} = \{1\}$ has cardinality 1, confirming $4 \in [2,7]$. Conversely, for $v = 1$ (binary '001'), $\mathcal{P}_v = \{0, 00, 001\}$ and $\mathcal{P}_v \cap \mathcal{B} = \emptyset$, so $1 \notin [2,7]$.

3.3 Detailed Construction of PrivDID

We now present the detailed construction of the PrivDID scheme, instantiating the abstract algorithms defined in Section 2.2.2. The public parameters $pp = (\mathbb{G}, q, G, H, \mathcal{H})$ are generated by $\mathsf{Setup}(1^\lambda)$ and used throughout, where $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ is a cryptographic hash function modeled as a random oracle.

3.3.1 Algorithms for Setup and Registration

These algorithms establish the basic cryptographic environment and user identity.

$\mathsf{Setup}(1^\lambda) \to pp$: Generate a description of a cyclic group $\mathbb{G}$ of prime order q with generators G and H, where H is derived in a verifiably random manner. Select a cryptographic hash function $\mathcal{H}$. Output $pp = (\mathbb{G}, q, G, H, \mathcal{H})$.
$\mathsf{CreateDIDD}(pp) \to (didd, pk, sk)$: Choose a random $sk \xleftarrow{\$} \mathbb{Z}_q$ as the private key. Compute the corresponding public key $pk = [sk]G$. Construct a standard DID Document $didd$ containing pk and other metadata. Output $(didd, pk, sk)$. The user registers $didd$ on the blockchain.

3.3.2 Algorithms for Credential Issuance

These algorithms allow a user to obtain a verifiable credential for an attribute value v.

$\mathsf{ApplyCred}(pp, v, aux) \to (\mathcal{C}, \pi_{\text{open}})$: Let m be the fixed bit length for representing the attribute (e.g., $m = 8$ for age). Compute the set of all binary prefixes of v: $\mathcal{P}_v = \mathsf{GetBinPrefixes}(v, m) = \{p_1, p_2, \ldots, p_m\}$ (see Section 3.2.4). For each prefix $p_i \in \mathcal{P}_v$:

1. Parse the auxiliary input aux to obtain or derive a random value $r_i \in \mathbb{Z}_q$.
2. Compute the Pedersen commitment to the prefix: $C_i = \mathsf{Com}(p_i; r_i) = [p_i]H + [r_i]G$.

The user's full **commitment** is $\mathcal{C} = \{C_1, \ldots, C_m\}$. The user then generates a non-interactive Schnorr proof π_{open} that demonstrates knowledge of the vector of random openings $\{r_i\}_{i=1}^m$ corresponding to the committed prefixes $\{p_i\}_{i=1}^m$. Formally, the proof is generated as:

$$\pi_{\text{open}} \leftarrow \mathsf{Prove}\left(\{r_i\}_{i=1}^m \;\middle|\; \{C_i, p_i\}_{i=1}^m : \bigwedge_{i=1}^m (C_i - [p_i]H = [r_i]G) \right).$$

Output $(\mathcal{C}, \pi_{\text{open}})$. The user stores $(\mathcal{C}, \{r_i\}, v, aux)$ locally.

$\mathsf{GenCred}(pp, didd, \text{attr}, v, \mathcal{C}, \pi_{\text{open}}, \mathsf{Proof}_v) \to VC$: This algorithm is executed by the Identity Provider (IdP).

1. Verify the external evidence Proof_v to cryptographically validate the truth of the claimed attribute value v. For example, if attr = "age" and $v = 20$, this step may involve verifying the digital signature on a government-issued electronic identity document that contains the birth date corresponding to age 20.

2. Verify the non-interactive Schnorr proof π_{open} against the commitment $\mathcal{C} = \{C_1, \ldots, C_m\}$ and the claimed prefixes $\{p_i\}_{i=1}^m = \text{GetBinPrefixes}(v, m)$. Formally, this verification checks:

   $$\mathsf{Verify}\left(\pi_{\text{open}},\ \{C_i, p_i\}_{i=1}^m : \bigwedge_{i=1}^m (C_i - [p_i]H = [r_i]G) \right) \stackrel{?}{=} 1.$$

   If the proof is invalid, return the error symbol $\perp$.

3. If both verifications succeed, the IdP creates a standard-compliant Verifiable Credential VC. The structure of VC includes, but is not limited to:

   - A unique identifier.
   - The DID of the holder and the issuer.
   - The attribute type (attr).
   - The **protected attribute value**: the commitment $\mathcal{C}$ (instead of the plaintext v).
   - A valid digital signature σ_{issuer} from the IdP, computed over the above contents.

4. The IdP submits the signed VC to the blockchain network. After consensus, VC is immutably anchored to the ledger $\mathcal{L}$, creating a permanent link between the user's DID and the attested commitment $\mathcal{C}$.

5. Return the issued VC to the user (or a success acknowledgment).

3.3.3 Algorithms for Service Access

These are the core algorithms enabling privacy-preserving service access. We describe the case for a generic interval predicate $\theta : v_l \le v \le v_h$; other forms (e.g., $v \ge v_l$, $v \le v_h$ or $v = v_d$) can be derived similarly.

$\mathsf{ApplyServ}(pp, v, aux, \theta, \mathcal{L}) \to (\pi_\theta, I)$: Given a predicate θ specifying an interval $[v_l, v_h]$, the user generates a zero-knowledge proof of inclusion.

1. **Encode Predicate:** Compute the minimal covering set for the interval $[v_l, v_h]$ using the function GetBRC:

   $$\mathcal{B} = \text{GetBRC}(v_l, v_h, m).$$

2. **Form Anonymity Set:** Let n_u be a privacy parameter chosen by the user, which determines the size of the anonymity set. From the public ledger $\mathcal{L}$, randomly select a set of $n_u - 1$ other valid and unrevoked commitments for the same attribute type (e.g., all are "age" commitments).

   Crucially, this selection is made without regard to the underlying attribute values committed. The only requirements are that the commitments are correctly formed and currently valid on the ledger. Combine these with the user's own commitment $\mathcal{C}_U$ (derived from (v, aux)) to form an initial list of n_u commitments. Then, apply a **random permutation** to this list to obtain the final, ordered anonymity set $\mathbb{A} = (\mathbb{A}[1], \mathbb{A}[2], \ldots, \mathbb{A}[n_u])$. Let I be the descriptor (e.g., a set of transaction identifiers) that uniquely identifies $\mathbb{A}$ within $\mathcal{L}$.

3. **Construct Public Key Matrix:** For each covering prefix $\text{brc}_i \in \mathcal{B}$ and for each index $j = 1, \ldots, n_u$, compute a derived public key:

   $$PK_{i,j} = C_{\ell_i}^{(j)} - [\text{brc}_i]H,$$

   where $\ell_i = \text{binLen}(\text{brc}_i)$, and $C_{\ell_i}^{(j)}$ denotes the specific prefix commitment of length ℓ_i within $\mathbb{A}[j]$. Organize all $PK_{i,j}$ into a matrix **PK**.

4. **Generate Ring Signature:** Let $\mathcal{P}_U = \mathsf{GetBinPrefixes}(v, m)$ be the set of all binary prefixes of the user's attribute value v. Compute the intersection $I^* = \mathcal{P}_U \cap \mathcal{B}$. By the property of GetBRC, I^* contains exactly one covering prefix, denoted brc^*. Let i^* be the index of brc^* in $\mathcal{B}$, and let j^* be the position of the user's own commitment $\mathcal{C}_U$ in the permuted anonymity set $\mathbb{A}$.

   The user's secret key for the proof is the randomness r^* (from aux) that corresponds to the commitment of brc^* in $\mathcal{C}_U$.

   $$PK_{i^*,j^*} = C_{\ell_{i^*}}^{(j^*)} - [\text{brc}^*]H = [r^*]G,$$

   where $\ell_{i^*} = \text{binLen}(\text{brc}^*)$ and $C_{\ell_{i^*}}^{(j^*)}$ is the commitment of length ℓ_{i^*} within $\mathcal{C}_U$.

   To generate the proof, the user constructs a ring

   $$\mathcal{R} = \left\{ PK_{i,j} \mid i = 1, \ldots, |\mathcal{B}|,\ j = 1, \ldots, n_u \right\},$$

i.e., the set of all derived public keys from matrix **PK**. Using r^* as the secret key for the public key $PK_{i^*,j^*} \in \mathcal{R}$, the user creates a ring signature σ on a service-related message *msg* (e.g., a hash of the access request). The resulting proof is $\pi_\theta = \sigma$. Output (π_θ, I).

$\mathsf{VerifyServ}(pp, \mathcal{L}, I, \theta, \pi_\theta) \to \{0,1\}$: The service provider verifies the proof, learning only that θ holds for some member of $\mathbb{A}$, with v and the prover's identity within $\mathbb{A}$ remaining hidden.

1. **Retrieve Data:** Using the descriptor I, retrieve the ordered anonymity set $\mathbb{A} = (\mathbb{A}[1], \ldots, \mathbb{A}[n_u])$ from the ledger $\mathcal{L}$.

2. **Reconstruct Matrix:** Recompute the public-key matrix **PK** identically to step 3 of $\mathsf{ApplyServ}$, using $\mathcal{B} = \mathsf{GetBRC}(v_l, v_h, m)$ and the full set $\mathbb{A}$.

3. **Build Ring:** Form the verification ring $\mathcal{R}$ as the set of all public keys in **PK**, i.e.,

$$\mathcal{R} = \left\{ PK_{i,j} \mid i = 1, \ldots, |\mathcal{B}|,\ j = 1, \ldots, n_u \right\}.$$

4. **Verify Signature:** Verify the ring signature π_θ against the ring $\mathcal{R}$ and the context message *msg*. If the verification succeeds, output 1 (grant access); otherwise, output 0.

This construction directly fulfills the interface defined in Section 2.2.2, leveraging the primitives from Section 3.2 to achieve both predicate verification and session unlinkability:

- **Predicate verification:** The ring signature proves knowledge of a secret key corresponding to one of the public keys derived from the covering set $\mathcal{B}$. A successful verification guarantees that the user's attribute satisfies θ.

- **Session unlinkability:** The ring signature ensures signer anonymity within the ring $\mathcal{R}$. Consequently, the verifier cannot link two proofs to the same credential, preventing cross-session tracking.

4 Security Analysis of the Proposed Scheme

In this section, we provide a formal security analysis of the PrivDID scheme described in Section 3. We demonstrate that the scheme satisfies the four security properties defined in Section 2.3: Completeness, Consistency, Attribute-Based Indistinguishability, and Session Unlinkability. All proofs are conducted in the random oracle model under standard cryptographic assumptions.

4.1 Completeness

Theorem 1 (Completeness of PrivDID). The PrivDID scheme is complete under the definitions in Section 2.3.1.

Proof. We consider the two aspects of completeness separately.

Credential generation completeness. Consider experiment $\mathsf{Exp}^{\text{VC-COMP}}_{\Pi,\mathcal{S}}(\lambda)$. The challenger $\mathscr{C}$, holding a valid tuple $(attr, v, aux, \mathsf{Proof}_v)$, executes $\mathsf{ApplyCred}$ to obtain $(\mathcal{C}, \pi_{\text{open}})$.

By construction, $\mathsf{ApplyCred}$ computes the correct $\mathcal{C}$ for v and generates a Schnorr proof π_{open} attesting to its correct formation. The simulator $\mathcal{S}$ (as IdP) then executes $\mathsf{GenCred}$ on the inputs $(attr, v, \mathcal{C}, \pi_{\text{open}}, \mathsf{Proof}_v)$.

By the perfect completeness of the Schnorr proof system, π_{open} verifies successfully. The external evidence Proof_v is valid by experiment setup. Therefore, $\mathsf{GenCred}$ outputs a valid VC with overwhelming probability.

Thus, for all PPT $\mathcal{S}$, the credential generation advantage $\mathsf{Adv}^{\text{VC}-\text{COMP}}_{\Pi,\mathcal{S}}(\lambda)$ is negligible. The scheme satisfies credential generation completeness.

Service access completeness. Consider experiment $\mathsf{Exp}^{\text{SERV-COMP}}_{\Pi,\mathcal{S}}(\lambda)$. The challenger $\mathscr{C}$, holding (v, aux, θ) where v satisfies θ and a ledger $\mathcal{L}$ containing a valid credential for v, executes $\mathsf{ApplyServ}$ to obtain a proof (π_θ, I).

By construction, $\mathsf{ApplyServ}$ encodes θ into $\mathcal{B} = \mathsf{GetBRC}(v_l, v_h, m)$, forms an anonymity set $\mathbb{A}$ containing the user's commitment $\mathcal{C}_U$, and constructs a public-key matrix **PK**. Since v satisfies θ, there exists a unique $\mathsf{brc}^* \in \mathcal{P}_U \cap \mathcal{B}$. Using the corresponding secret r^*, the algorithm generates a ring signature $\sigma = \pi_\theta$ over the ring $\mathcal{R}$ derived from **PK**.

The verifier in $\mathsf{VerifyServ}$ reconstructs the identical **PK** and $\mathcal{R}$ from $\mathcal{L}$ and I. By the perfect completeness of the underlying ring signature scheme, σ verifies successfully. Thus, $\mathsf{VerifyServ}$ outputs 1 with overwhelming probability.

Therefore, for all PPT $\mathcal{S}$, $\mathsf{Adv}^{\text{SERV}-\text{COMP}}_{\Pi,\mathcal{S}}(\lambda)$ is negligible. The scheme satisfies service access completeness.

Combining both parts, we conclude that PrivDID satisfies the completeness property. □

4.2 Consistency

Theorem 2 (Consistency of PrivDID). Assume the discrete logarithm problem is hard in the group $\mathbb{G}$. Then PrivDID satisfies the consistency property defined in Section 2.3.3.

Proof (Sketch). Assume a PPT adversary $\mathcal{A}$ breaks consistency by outputting $(v, v', \mathcal{C}, \pi_{\text{open}}, \pi'_{\text{open}})$ with $v \neq v'$ and both openings valid. These are non-interactive Schnorr proofs of knowledge for the random openings of the Pedersen commitments. By the knowledge-soundness of Schnorr proofs in the random-oracle model [40], we can extract the underlying randomness $\{r_i\}$ and $\{r'_i\}$ that satisfy the respective openings for v and v'.

Since $v \neq v'$, their binary prefixes differ at some index j, i.e., $p_j \neq p'_j$. From the verification equations for the j-th

commitment $C_j \in \mathcal{C}$,

$$[p_j]H + [r_j]G = [p'_j]H + [r'_j]G,$$

we obtain $[p_j - p'_j]H = [r'_j - r_j]G$. Because $p_j \neq p'_j \pmod q$, we can compute

$$H = \left[\frac{r'_j - r_j}{p_j - p'_j}\right] G,$$

which gives the discrete logarithm of H to base G. Therefore any adversary that wins the consistency game can be used to solve the DL problem in $\mathbb{G}$; under the DL assumption,

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

□

4.3 Attribute-Based Indistinguishability

Theorem 3 (Attribute-Based Indistinguishability of PrivDID). Assuming the perfect hiding property of Pedersen commitments, the PrivDID scheme satisfies the Attribute-Based Indistinguishability property defined in Section 2.3.4.

Proof (Sketch). The proof follows from the **perfect hiding** property of Pedersen commitments. In the indistinguishability experiment, the adversary receives a challenge commitment $\hat{C}_b = \{C_1, \ldots, C_m\}$ where each $C_i = [p_i]H + [r_i]G$, with p_i a prefix of the chosen attribute v_b and r_i uniform. For any prefix value p'_i, there exists a randomness r'_i such that $C_i = [p'_i]H + [r'_i]G$; hence the distribution of each C_i is identical for all possible prefixes. Because the m commitments are generated with *independent randomness* $\{r_i\}$, the joint distribution of the whole commitment $\hat{C}_b$ is identical for any two distinct attribute values v_0, v_1. Thus even an unbounded adversary cannot guess b better than random:

$$\Pr\left[\mathsf{Exp}^{\mathsf{AIND}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] = \frac{1}{2}, \qquad \mathsf{Adv}^{\mathsf{AIND}}_{\Pi,\mathcal{A}}(\lambda) = 0.$$

□

4.4 Session Unlinkability

Theorem 4 (Session Unlinkability of PrivDID). Assume the ring signature scheme used in PrivDID provides statistical anonymity. Then the PrivDID scheme satisfies the unlinkability property defined in Section 2.3.5.

Proof. Consider the unlinkability experiment $\mathsf{Exp}^{\mathsf{SUNL}}_{\Pi,\mathcal{A}}(\lambda)$. The adversary $\mathcal{A}$ selects two credentials VC_1 and VC_2 corresponding to attribute values v_1 and v_2, both satisfying a predicate θ. The challenger $\mathcal{C}$ then generates two proofs (π_1, I_1) and (π_2, I_2) according to the random bit b:

- If $b = 0$, both proofs are generated from VC_1 (same credential).
- If $b = 1$, π_1 is from VC_1 and π_2 from VC_2 (different credentials).

In both cases, each proof π_i is a ring signature σ_i constructed over the ring $\mathcal{R}_i$ derived from the public-key matrix $\mathbf{PK}_i$. The anonymity set for each proof includes both VC_1 and VC_2 (as required in step 5 of the unlinkability experiment), ensuring that the rings $\mathcal{R}_1$ and $\mathcal{R}_2$ contain public keys corresponding to both credentials.

By the **statistical anonymity** property of the ring signature scheme, given a valid signature σ, the probability that an adversary can identify the true signer within the ring is at most $1/|\mathcal{R}| + \mathsf{negl}(\lambda)$. Consequently, even an unbounded adversary cannot distinguish whether σ_1 and σ_2 were generated by the same signer (using the same secret key r^*) or by two different signers (using different secret keys r_1^* and r_2^*).

Therefore, for any PPT adversary $\mathcal{A}$,

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

Thus, PrivDID achieves session unlinkability under the anonymity guarantee of the underlying ring signature. □

4.5 Practical Security and System Considerations

Our formal analysis (Sections 4.1–4.4) establishes the core cryptographic guarantees of PrivDID. To complete the security analysis for real-world deployment, we address several critical system-level considerations.

Authentication via Unified Proof. The ApplyServ phase requires the service to verify the user's legitimacy and their attribute compliance (e.g., age ≥ 18). Using the long-term DID public key directly would authenticate the user but also link all their sessions. PrivDID avoids this by employing a single ring signature that simultaneously proves two facts in zero-knowledge: (1) the signer holds the private key corresponding to a **specific, unrevoked commitment within the presented anonymity set**, and (2) that committed value satisfies the service's BRC-encoded policy (compliance). This unified proof thus authenticates the user and validates their hidden attribute in one unlinkable step.

Revocation. Revocation is managed through the standard DID revocation mechanism, where the Identity Provider (IdP) submits an on-chain transaction to revoke a credential. During the ApplyServ phase, users must construct the anonymity set exclusively from active (unrevoked) commitments as recorded on the ledger. The service provider verifies this revocation status before validating the ring signature, ensuring that any proof depending on a revoked commitment is rejected.

Trust in the Identity Provider. Our scheme is based on a single, well-defined trust assumption regarding privacy: *the IdP will not maliciously disclose the user's private attributes it learns during credential issuance*. Crucially, because the user's submission in ApplyCred includes only a zero-knowledge proof π_{open}—which reveals no information about the opening randomness $\{r_i\}$—the IdP cannot impersonate the user. All other critical actions, such as credential issuance and revocation, are recorded on-chain, making them publicly verifiable and non-repudiable. This design minimizes the trust footprint and ensures that any misuse by the IdP can at least be detected and proven after the fact, providing accountability.

The off-chain ApplyCred phase allows the IdP to deny service, a common limitation in interactive issuance. Mitigations like economic bonds or fair exchange are possible.

Integration with External Security Mechanisms. For practical deployment, the core protocol is designed to be augmented through a layered integration of external security mechanisms. At the application and cryptographic layer, standard defenses such as replay protection (via challenge nonces) and external auditing (leveraging on-chain verifiability) can be seamlessly integrated to enhance practical security without compromising our scheme's privacy guarantees. Importantly, these guarantees operate at a layer orthogonal to network-level threats like traffic analysis or metadata linkage. Consequently, to address this separate class of threats, the system architecture should integrate complementary network-layer protections, such as anonymity networks (e.g., Tor) or communication mixing, forming a comprehensive defense-in-depth strategy.

5 Performance Evaluation

We evaluate the performance of PrivDID from both theoretical and experimental perspectives. In this section, we first present a comparative theoretical analysis with existing privacy-preserving DID schemes, comparing their privacy guarantees and cryptographic foundations. Subsequently, we report empirical results based on a prototype implementation, demonstrating the practical feasibility of PrivDID in real-world settings.

5.1 Theoretical Analysis and Comparison

We compare PrivDID with three representative privacy-enhancing DID schemes (Table 1).

The hash-based approach of [17] conceals the link between attributes and user identities on the ledger. However, this privacy guarantee collapses during active use, as any predicate check requires the user to reveal the entire attribute value to the verifier. [18] stores explicit predicate statements (context-based credentials) on-chain, enabling verification but directly leaking the predicates themselves—a form of partial disclosure. This approach provides unlinkability only across different context-based credentials (i.e., different predicates cannot be linked to the same user), but fails to provide session unlinkability: multiple uses of the same context-based credential remain linkable, enabling longitudinal tracking. Moreover, supporting multiple application scenarios (e.g., proving age ≥ 18 for voting and age < 60 for employment) requires issuing separate credentials for each predicate, introducing redundancy and system inefficiency. In contrast, both [24] and our PrivDID employ Pedersen commitments that hide the attribute completely, while supporting efficient zero-knowledge proofs that reveal only the truth value of the predicate, thereby achieving true minimal disclosure.

Table 1: Comparison of privacy-enhancing DID schemes.

Scheme	Minimal Disclosure	Compatibility	Trusted Setup	Unlinkability
[17]	No	No	Yes	No
[18]	limited	No	Yes	limited
[24]	Yes	Full	No	No
PrivDID	Yes	Full	No	Yes

Both [17] and [18] rely on zk-SNARKs [41,42] that require complex trusted-setup ceremonies—a significant drawback in decentralized environments where reliable third parties are scarce. In contrast, PrivDID (like [24]) uses Pedersen commitments and ring-signature-based proofs that need no trusted initialization ceremony. This approach demonstrates superior computational and storage efficiency while eliminating a critical trust assumption.

Crucially, PrivDID achieves these advances without compromising the full W3C compatibility essential for adoption, a property shared by [24] but not guaranteed by [17, 18]. Most distinctively, PrivDID introduces session unlinkability—a guarantee absent from all prior schemes. By integrating ring signatures with prefix-based predicate proofs, our scheme ensures that multiple presentations of the same credential cannot be correlated, thereby thwarting longitudinal tracking while preserving minimal disclosure..

In summary, PrivDID is the first fully decentralized and publicly verifiable scheme that achieves both *minimal disclosure* and *session unlinkability*. It does so by extending privacy foundation of [24] with provable unlinkability, while retaining full W3C compatibility, efficient predicate verification, and no requirement for a trusted setup.

5.2 Experimental Evaluation

Implementation Details. Our prototype of PrivDID is implemented in C (C11 standard) and leverages the OpenSSL library for elliptic curve operations over the `secp256k1` curve. For the ring signature component, we employ a variant of

the algorithm from Monero [43], adapted for our setting by omitting the components related to the key image. This modification removes the double-spending prevention mechanism, which is irrelevant for credential presentation, while fully preserving the anonymity and unforgeability guarantees that are essential for our security analysis (Section 4). All hash operations use SHA-256 as the underlying primitive.

Experimental Setup. The code was compiled with GCC 11.4. All experiments were executed on a server equipped with an Intel Xeon Platinum 8358P processor (2.60 GHz) and 1 TB of RAM, running Ubuntu Linux 22.04. To accurately measure the intrinsic computational cost of the cryptographic algorithms, our implementation is single-threaded; this isolates the baseline performance of the core operations. Further speed-up via parallelization, algorithm-specific optimizations, or hardware acceleration is orthogonal to the security analysis and is left for future work.

Parameters. In the following scalability analysis, we examine three key parameters: the attribute bit-length m, the anonymity-set size n_u, and the size $|\text{brc}|$ (i.e., $|\mathcal{B}|$) of the BRC set that encodes the predicate.

Prefix Encoding. We apply a fixed encoding to all binary prefixes before commitment: '0'→'01', '1'→'10'. This encoding is consistently applied to both attribute prefixes in user commitments and BRC nodes during proof generation and verification, ensuring unambiguous matching. Since these encoded values are used as scalars in elliptic-curve multiplications, whose results are fixed-size points, the encoding adds no storage or communication overhead.

5.2.1 Space Overhead

We analyze the space requirements of PrivDID, considering both on-chain storage and communication payloads. The implementation uses compressed `secp256k1` points (33 bytes) and 32-byte scalars.

On-chain storage per credential. Each credential stores m Pedersen commitments (one per binary prefix), requiring: $\text{Size}_{\text{chain}} = m \times 33$ bytes.

ApplyCred payload. To obtain a credential, the user sends the attribute value v ($\lceil m/8 \rceil$ bytes), the commitment $\mathcal{C}$ ($m \times 33$ bytes), and a Schnorr proof π_{open} ($m \times 65$ bytes). The Identity Provider computes all binary prefixes of v locally to verify the proof. Thus: $\text{Size}_{\text{ApplyCred}} = \lceil m/8 \rceil + m \times 98$ bytes.

ApplyServ payload. For service access, the user generates a ring signature over a ring of size $|\text{brc}| \times n_u$. The signature consists of three components per ring member: a curve point (33 bytes), a response scalar (32 bytes), and a challenge scalar (32 bytes), along with an additional n_u transaction identifiers (32 bytes each). Thus: $\text{Size}_{\text{ApplyServ}} = |\text{brc}| \times n_u \times (33+32+32) + n_u \times 32 = n_u(97|\text{brc}|+32)$ bytes.

Example. With $m = 32$, $|\text{brc}| = 4$, $n_u = 10$: On-chain: 1.03 KB; ApplyCred: 3.07 KB; ApplyServ: 4.10 KB. These modest overheads confirm that PrivDID is practical for real-world deployment.

5.2.2 Performance of Credential Issuance Phases

The first experiment evaluates the overhead of credential issuance. We measure the execution time of the ApplyCred and GenCred algorithms while varying the bit length m from 8 to 64.

Figure 3 shows the execution time of ApplyCred. The time increases linearly with m, as expected, since the user must compute m independent Pedersen commitments and generate a Schnorr proof of knowledge for all opening randomnesses.

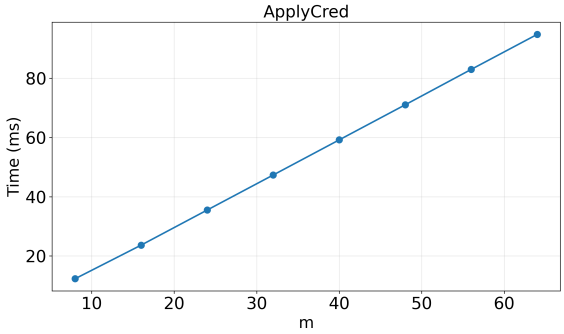


Figure 3: Execution time of ApplyCred

Figure 4 presents the corresponding time for GenCred, executed by the Identity Provider (IdP). The GenCred algorithm also scales linearly with m, as the IdP must verify the user's Schnorr proof. For $m = 32$, both ApplyCred (≈47 ms) and GenCred (≈36 ms) complete in tens of milliseconds, confirming the practical efficiency of the credential setup phases in real-world scenarios.

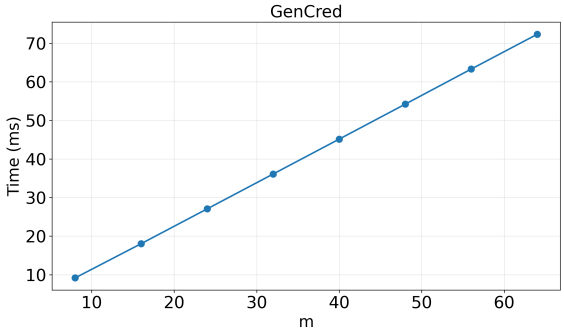


Figure 4: Execution time of GenCred

5.2.3 Performance of Service Access and Verification Phases

The second experiment measures the overhead of the privacy-preserving service access protocol, i.e., ApplyServ and VerifyServ. We fixed $m = 32$ and varied two parameters: (1) n_u

from 5 to 40, and (2) $|\text{brc}| \in \{4, 8, 12\}$, which determines the predicate complexity.

Figure 5 depicts the execution time of ApplyServ. The time grows linearly with both n_u and $|\text{brc}|$, because the user must construct a public-key matrix of size $|\text{brc}| \times n_u$ and generate a ring signature over the corresponding ring. Even for a moderately large anonymity set ($n_u = 40$) and a predicate with $|\text{brc}| = 12$, ApplyServ completes in under 550 ms.

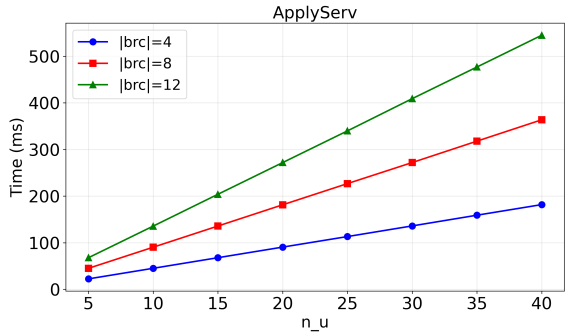


Figure 5: Execution time of ApplyServ

Figure 6 shows the verification time of VerifyServ. The verifier reconstructs the same public-key matrix and checks the ring signature, leading to a similar linear trend. The overhead of VerifyServ closely matches that of ApplyServ, with times staying below 550 ms for the same parameter ranges.

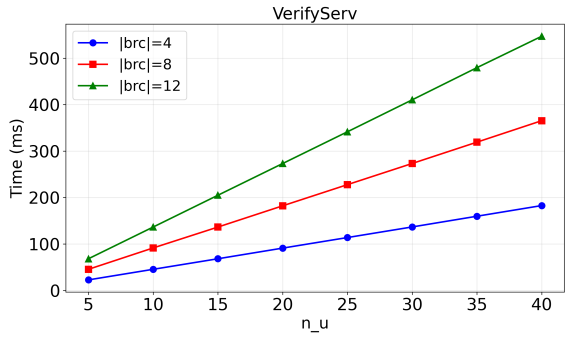


Figure 6: Execution time of VerifyServ

5.2.4 Summary

The experimental results validate the practical efficiency of PrivDID. The Credential Issuance Phases complete within tens of milliseconds for typical attribute lengths. The Service Access and Verification Phases scale linearly with the anonymity set size and predicate complexity, yet remain within 550 ms even for moderately large configurations. These findings demonstrate that PrivDID achieves strong privacy guarantees—minimal disclosure and session unlinkability—without imposing prohibitive computational overhead, making it suitable for deployment in real-world decentralized identity systems.

6 Conclusion

This paper presents a cryptographic framework for decentralized identity that reconciles strong privacy with public verifiability. The framework unifies two complementary privacy goals: *minimal disclosure* and *session unlinkability*. We instantiate this framework as PrivDID—a concrete, W3C-compatible scheme that requires no trusted setup. PrivDID achieves both goals through a single ring signature, which proves in zero-knowledge that a set of Pedersen commitments (encoding the attribute's binary prefixes) satisfies a range predicate expressed via BRC encoding, while inherently providing session unlinkability.

Formal analysis shows that PrivDID provably guarantees attribute privacy and session unlinkability under standard assumptions. A prototype implementation confirms its practicality: all operations complete within hundreds of milliseconds, and space/communication overhead scales gracefully with system parameters (attribute dimension, anonymity set, and predicate complexity).

While the current instantiation uses `secp256k1` and a simplified ring signature as a proof-of-concept, the framework is designed to be adaptable. Future work may explore different cryptographic backends (e.g., Ristretto25519) and richer predicate logic.

Overall, the framework decouples privacy goals from concrete primitives, offering a blueprint for future verifiable and user-sovereign identity systems that can evolve without sacrificing privacy.

References

[1] Avellaneda O, Bachmann A, Barbir A, et al. Decentralized identity: Where did it come from and where is it going?. IEEE Communications Standards Magazine, 2019, 3(4): 10-13.

[2] Dib O, Rababah B. Decentralized identity systems: Architecture, challenges, solutions and future directions. Annals of Emerging Technologies in Computing (AETiC), 2020, 4(5): 19-40.

[3] Fdhila W, Stifter N, Kostal K, et al. Methods for decentralized identities: Evaluation and insights//International Conference on Business Process Management. Cham: Springer International Publishing, 2021: 119-135.

[4] Stockburger L, Kokosioulis G, Mukkamala A, et al. Blockchain-enabled decentralized identity management: The case of self-sovereign identity in public transportation. Blockchain: Research and Applications, 2021, 2(2): 100014.

[5] Luecking M, Fries C, Lamberti R, et al. Decentralized identity and trust management framework for Internet of Things//2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2020: 1-9.

[6] Zhou L, Diro A, Saini A, et al. Leveraging zero knowledge proofs for blockchain-based identity sharing: A survey of advancements, challenges and opportunities. Journal of Information Security and Applications, 2024, 80: 103678.

[7] Ahmed M R, Islam A K M M, Shatabda S, et al. Blockchain-based identity management system and self-sovereign identity ecosystem: A comprehensive survey. IEEE Access, 2022, 10: 113436-113481.

[8] Zheng Z, Xie S, Dai H N, et al. Blockchain challenges and opportunities: A survey. International journal of web and grid services, 2018, 14(4): 352-375.

[9] Nofer M, Gomber P, Hinz O, et al. Blockchain. Business & information systems engineering, 2017, 59(3): 183-187.

[10] Di Pierro M. What is the blockchain?. Computing in Science & Engineering, 2017, 19(5): 92-95.

[11] Yaga D, Mell P, Roby N, et al. Blockchain technology overview. arXiv preprint arXiv:1906.11078, 2019.

[12] Dinh T T A, Liu R, Zhang M, et al. Untangling blockchain: A data processing view of blockchain systems. IEEE transactions on knowledge and data engineering, 2018, 30(7): 1366-1385.

[13] Guo H, Yu X. A survey on blockchain technology and its security. Blockchain: research and applications, 2022, 3(2): 100067.

[14] Dunphy P, Petitcolas F A P. A first look at identity management schemes on the blockchain. IEEE security & privacy, 2018, 16(4): 20-29.

[15] Kuperberg M. Blockchain-based identity management: A survey from the enterprise and ecosystem perspective. IEEE Transactions on Engineering Management, 2019, 67(4): 1008-1027.

[16] https://www.w3.org/TR/did-core/

[17] Yang X, Li W. A zero-knowledge-proof-based digital identity management scheme in blockchain. Computers & Security, 2020, 99: 102050.

[18] Maram D, Malvai H, Zhang F, et al. Candid: Can-do decentralized identity with legacy compatibility, sybil-resistance, and accountability//2021 IEEE Symposium on Security and Privacy (SP). IEEE, 2021: 1348-1366.

[19] Voigt P, Von dem Bussche A. The eu general data protection regulation (gdpr). A practical guide, 1st ed., Cham: Springer International Publishing, 2017

[20] Zaeem R N, Barber K S. The effect of the GDPR on privacy policies: Recent progress and future promise. ACM Transactions on Management Information Systems (TMIS), 2020, 12(1): 1-20.

[21] Albrecht J P. How the GDPR will change the world. Eur. Data Prot. L. Rev., 2016, 2: 287.

[22] ISO/IEC. (2022). Information security, cybersecurity and privacy protection—Information security management systems—Requirements (ISO/IEC 27001:2022). International Organization for Standardization.

[23] National Institute of Standards and Technology (NIST). (2020). NIST privacy framework: A tool for improving privacy through enterprise risk management (Version 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.CSWP.01162020

[24] https://eprint.iacr.org/2026/077

[25] Goldwasser S., Micali S., Rackoff C. Knowledge complexity of interactive proofs //Proceedings of the seventeenth annual ACM symposium on Theory of computing. 1985: 291-304.

[26] Groth J. On the size of pairing-based non-interactive arguments//Annual international conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016: 305-326.

[27] Sun X, Yu F R, Zhang P, et al. A survey on zero-knowledge proof in blockchain. IEEE network, 2021, 35(4): 198-205.

[28] Noether S, Mackenzie A. Ring confidential transactions. Ledger, 2016, 1: 1-18.

[29] Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing//Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991: 129-140.

[30] Damgård I. Commitment schemes and zero-knowledge protocols//School organized by the European Educational Forum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998: 63-86.

[31] Hao, Feng. Schnorr non-interactive zero-knowledge proof. No. rfc8235. 2017.

[32] Camenisch, J., Kiayias, A., Yung, M. (2009). On the Portability of Generalized Schnorr Proofs. In: Joux, A. (eds) Advances in Cryptology - EUROCRYPT 2009.

[33] Bender A, Katz J, Morselli R. Ring signatures: Stronger definitions, and constructions without random oracles//Theory of Cryptography Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006: 60-79.

[34] Rivest, R.L., Shamir, A., Tauman, Y. (2006). How to Leak a Secret: Theory and Applications of Ring Signatures. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds) Theoretical Computer Science. Lecture Notes in Computer Science, vol 3895. Springer, Berlin, Heidelberg.

[35] Wang L, Zhang G, Ma C. A survey of ring signature. Frontiers of Electrical and Electronic Engineering in China, 2008, 3(1): 10-19.

[36] Demertzis, S.Papaopoulos, O.Papapetrou, A.Deligian nakis, and M.N. Garofalakis. Practical private range search revisited. In SIGMOD, 2016.

[37] Zuo, C. , et al. Dynamic Searchable Symmetric Encryption Schemes Supporting Range Queries with Forward/Backward Privacy. In ESORICS 2018.

[38] WangJiafan, ChowSherman. Forward and Backward-Secure Range-Searchable Symmetric Encryption. Proceedings on Privacy Enhancing Technologies, 2022(1):28-48.

[39] Kiayias A , Papadopoulos S , Triandopoulos N , et al. Delegatable pseudorandom functions and applications. ACM CCS2013, pp.669–684

[40] Pointcheval D, Stern J. Security proofs for signature schemes[C]//International conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996: 387-398.

[41] Groth J, Maller M. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs//Annual International Cryptology Conference. Cham: Springer International Publishing, 2017: 581-612.

[42] Eberhardt J, Tai S. Zokrates-scalable privacy-preserving off-chain computations//2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2018: 1084-1091.

[43] https://www.getmonero.org/resources/research-lab/pubs/cryptonote-whitepaper.pdf