A New Generation of Fast and Flexible Anonymous Credential Primitives for Private Digital Identity

A thesis submitted in fulfilment of the requirements for the degree of Master of Philosophy

Samuel Polgar

Faculty of Engineering

The University of Sydney

2025

Chapter 1

*

Table of Contents

List of Figures

Chapter 2

Introduction

(To be written properly) The thesis focuses on improving Anonymous Credentials and specifically for a few use-cases. We will soon be using digital credential wallets (mandated by the EU) [Eur24] and similar initiatives globally.

2.0.1 Chapter 3

Improved Constructions of Anonymous Credentials from New Rerandomizable Signatures. This chapter is motivated by the proliferation of credential wallets and needing to (quickly and privately) verify credentials. In this chapter I formalize security for a variant of the PS signature [PS16], from the UTT paper [TBA+22], I improve it by making it more efficient and improving security (against malicious issuers) and empirically show it is best in class with concrete benchmarks.

2.0.2 Chapter 4

Identity Binding Multi Issuer Multi Credential Anonymous Credentials. This chapter is motivated by thinking about what's required from credential wallet architecture to verify multiple credentials together. E.g., you have a driver's license, passport, and bank information in your credential wallet, and need to verify it together and prove it's from the same person. I create the security property "Identity Binding" for this situation. I show benchmarks for two different scenarios, 1) the cost of privacy, 2) the improvement from single issuer. 1) I compare the two scenarios, verifying credentials without privacy and with privacy and show that privacy preserving credentials doesn't cost much more (in overhead), not what it used to. 2) Our anonymous credential from 3 can be aggregated if signed with the same key. I show the efficiency benefit if this is the case.

2.0.3 Chapter 5

New Nullifier Constructions from the q-DDHI and Applications to Accountable Privacy Systems. A nullifier is a cryptographic object, similar in thought to a signature. It's created by someone, contains a user's secret key and a specific message, and is given to a verifier to prove that the user made the nullifier with a specific message and they own the key public key that did it. A user who spends their coin would make a nullifier from their secret key and the coin's serial number (like in zcash) and give it the bank to be able to spend it. In Anonymous Credentials, a nullifier ensures a user who requests a credential can't request another one, preventing sybil attacks, which is difficult to do when credentials are private and rerandomizable. In accountable privacy, nullifiers should be private, so they shouldn't share the public key. I create New Nullifiers from a series of zero-knowledge proof protocols I made. Starting with a non-private one but a faster version of a VRF [DY05], and then two nullifier constructions. We will use this for Sybil-resistant anonymous credentials. We show ours are super quick and simple compared to the current.

2.0.4 Chapter ??

Sybil Resistant Threshold Issued Identity System from Anonymous Credentials and Anonymous Nullifiers. In this chapter I add threshold-issuance for credentials and build a sybil-resistant private threshold-issued identity system drawing on the work from **??** and show it's way more efficient than the other state of the art.

2.0.5 Chapter 6

Anonymous Credentials Open Source Library - the literature and industry has a gap in implementations for gathering evaluation data for many of these schemes and thus it's hard to determine whether a new scheme has made an improvement or is just using a different technique. I built a few schemes and found many small tricks in the cryptography library make a big difference in concrete implementation efficiency, probably more difference than many "new" papers make theoretically. I wanted to represent this in the Thesis and open-source that for people to use and see.

[DCMMH22]

Future Work

Chapter 3

Improved Constructions of Anonymous Credentials from New Rerandomizable Signatures

3.0.6 Motivation

Digital interactions are shifting toward the self-sovereign identity (SSI) model, where users control cryptographic credentials in digital wallets. In this paradigm, a wallet is not just a storage container, but a user-controlled agent for issuing, holding, and presenting zero-knowledge proofs of credentials across interoperable systems. Governments and industry are mandating digital wallets (e.g., EU Digital Wallet Regulation [noa24]; 13 U.S. jurisdictions issuing mobile driver's licenses [AAM]). Credential Oracles [ZMM+20, CDH+25] are "feeder services" that generate verifiable data attestations from different sources such as Web 2.0 credentials, bank data, or offline information. The digital attestations are associated with a user's credential wallet and stored "within" the wallet. Commodity devices—from smartphones to IoT sensors—will sign outputs (photos, logs, transactions) with device keys (e.g., C2PA [C2P24]), turning everyday interactions into verifiable claims. As AI-generated media and deepfakes proliferate, cryptographic provenance becomes critical, exponentially increasing the volume and heterogeneity of credentials that wallets must manage.

3.0.7 Problems and Scope

The proliferation of credential types, sources, and volume reveals three gaps in existing schemes:

1. **Scalability for wallet-centric architectures.** In the SSI model, wallets will accumulate potentially thousands of heterogeneous credentials (identity documents, transaction receipts, device-signed content, oracle attestations). Presentation must remain succinct even in scenarios where proof aggregation is possible.

2. **Security against untrusted issuers.** The decentralized ecosystem includes credential oracles, device-embedded signers, and third-party attesters with varying security assumptions. User privacy (unlinkability, anonymity) must hold even when issuers collude or behave arbitrarily, protecting users from tracking by malicious ecosystem participants.

3. **Expressive, efficient zero-knowledge proofs.** Users need to prove dynamic policies over attribute sets (e.g., "prove I'm over 18 AND in EU AND my device captured this photo today"). Achieving rich predicate support beyond simple possession proofs, without heavy SNARK primitives, is essential for mobile device feasibility.

Solving these challenges requires a foundational understanding of how anonymous credentials evolved to address similar scalability and security concerns. While existing systems have made

significant progress, none fully address the demands of the SSI wallet-centric ecosystem we've described. To build this next-generation system, we must first examine the architectural decisions and cryptographic innovations that brought us to this point.

Anonymous credentials are a fundamental building block for privacy-preserving digital identity, aligning perfectly with the SSI principle that users should prove attributes without revealing full identity. Since their first inception by Chaum [Cha81], they have evolved from theoretical constructs to practical systems deployed in real-world applications such as U-Prove, Idemix, and PrivacyPass [CVH02, CKL+16], but none have been designed explicitly for wallet-centric architectures managing thousands of heterogeneous credentials.

Despite these advances, current ABC systems face critical gaps that prevent wide-scale deployment in the SSI ecosystem we've outlined. Systems like SPS-EQ [FHS19, CLPK22] and ACT [BRS23] achieve high efficiency but support only limited predicates, while approaches based on zkSNARKs enable rich expressiveness [RWGM22] but at computational cost. Furthermore, many anonymous credential systems assume honest issuers, leaving users vulnerable to privacy breaches from malicious credential providers, especially when issuers might be credential oracles.

This chapter develops the cryptographic foundations needed for credential wallets. We address each of the identified challenges through four key contributions that together enable efficient, secure, and expressive anonymous credentials at the scale demanded by the SSI ecosystem:

1. **Complete Security Proofs for an efficient PS signature variant from UTT:** We provide formal security proofs that were only sketched in [TBA+22]:
   - Position-binding vector commitments under SDLP in the Algebraic Group Model (Section 3.1.3)
   - EUF-CMA security for rerandomizable signatures over commitments under PS-LRSW (Section 3.2)
   - Satisfaction of Anonymous Credential model [FHS19] security properties—correctness, unforgeability, and anonymity—for multi-show operations (Section 3.5)
2. **Fastest Show+Verify Operations:** Our construction achieves 10-16% faster verification than UTT [TBA+22] and outperforms the most popular BBS+ variant [CDL16] - we validate through comprehensive benchmarks 3.6.
3. **Malicious Issuer Security through Key Verification Protocols:** We formalize anonymity guarantees against colluding issuers via mandatory key verification (Section 3.3.2), preventing deanonymization attacks through maliciously generated public keys—a vulnerability that existing schemes do not address.
4. **Empirical Validation of Schnorr Protocol Efficiency:** Comprehensive benchmarks 21reveal that Schnorr protocols achieve sublinear complexity for attribute proofs via multi-scalar multiplication, outperforming theoretical predictions and enabling almost constant-size expressive predicate evaluation.

Chapter Roadmap This chapter develops cryptographic foundations for efficient anonymous credentials in decentralized identity systems. Section 3.1 establishes our mathematical setting with SDLP and PS-LRSW assumptions.

Section 3.2 provides the first complete security proofs for our rerandomizable signature scheme, proving position-binding under SDLP and EUF-CMA security under PS-LRSW. Section 3.3 presents our key optimization: moving signatures from $\mathbb{G}_1$ to $\mathbb{G}_2$ for 10-16% faster verification, plus malicious issuer protection through key verification.

Section 3.4 demonstrates how Sigma protocols achieve efficient predicate evaluation over complex policies. Section 3.5 constructs our complete credential system with security proofs. Section

3.6 validates our approach through benchmarks, showing state-of-the-art performance with up to 83% verification speedup.

3.1 Preliminaries

3.1.1 Assumptions

Definition 3.1 (Symmetric Discrete Logarithm Assumption (SDLP)). *For any PPT adversary $\mathcal{A}$, we say the SDLP assumption holds if there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{l|l} \mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda) & \\ x \leftarrow\!\$\ \mathbb{Z}_p & : x = x' \\ x' \leftarrow\!\$\ \mathcal{A}(\mathsf{BG}, g^x, \tilde{g}^x) & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where validity of input can be verified by checking $e(g, \tilde{g}^x) = e(g^x, \tilde{g})$.

Definition 3.2 (Type-3 PS-LRSW Assumption). *For any PPT adversary $\mathcal{A}$, we say the Type-3 PS-LRSW assumption holds in the generic bilinear group model if there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{l|l} \mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda) & \\ x, y \leftarrow\!\$\ \mathbb{Z}_p, X \leftarrow g^x, \tilde{X} \leftarrow \tilde{g}^x, \tilde{Y} \leftarrow \tilde{g}^y & : \begin{array}{l} m^* \notin Q \wedge P_1 \neq 1_{\mathbb{G}_1} \wedge \\ e(P_1, \tilde{X} \cdot \tilde{Y}^{m^*}) = e(P_2, \tilde{g}) \end{array} \\ (m^*, P_1, P_2) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{x,y}}(\mathsf{BG}, X, \tilde{Y}) & \end{array}\right] \leq \mathsf{negl}$$

where Q is the set of queries made to $\mathcal{O}_{x,y}$ with access to x, y is defined by:

$$\mathit{Oracle}\ \mathcal{O}_{x,y}(m):\ \mathit{Samples}\ h \leftarrow\!\$\ \mathbb{G}_1,\ \mathit{Returns\ the\ pair}\ P = (h, h^{x+my})$$

3.1.2 Dual-Group Pedersen Commitments

In this section, we introduce a specialized extension of Pedersen commitments that supports vector messages, rerandomizability, and position binding. Our main contribution is a security proof in the Algebraic Group Model (AGM) that establishes position binding based on the Symmetric Discrete Logarithm Problem (SDLP).

Definition 3.3 (Commitment Scheme). *A commitment scheme* Com *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$*: is a probabilistic algorithm that takes as input a security parameter λ and outputs a commitment key* ck*. The message space $\mathcal{M}$ is implicitly defined by* ck*.*
- $\mathsf{Commit}(\mathsf{ck}, m) \to (\mathsf{cm}, r)$*: is a probabilistic algorithm that takes as input the commitment key* ck *and a message $m \in \mathcal{M}$. It outputs a commitment* cm *and an opening value r.*
- $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, m, r) \to b$*: is a deterministic algorithm that takes as input the commitment key* ck*, a commitment* cm*, a message m, and an opening value r. It outputs a bit $b \in \{0, 1\}$, where 1 indicates a valid opening and 0 indicates an invalid opening.*

Definition 3.4 (Correctness). *A commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is correct if for all $\lambda \in \mathbb{N}$, all commitment keys $\mathsf{ck} \in [\mathsf{Setup}(1^\lambda)]$, and all messages $m \in \mathcal{M}$:*

$$\Pr[(\mathsf{cm}, r) \leftarrow\!\$\ \mathsf{Commit}(\mathsf{ck}, m) : \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, m, r) = 1] = 1.$$

Definition 3.5 (Hiding). *A commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is hiding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$\left|\Pr\left[\begin{array}{l|l} \mathsf{ck} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) & \\ (m_0, m_1) \leftarrow\!\$\ \mathcal{A}(\mathsf{ck}) & \\ b \leftarrow\!\$\ \{0, 1\} & : b' = b \\ (\mathsf{cm}, r) \leftarrow\!\$\ \mathsf{Commit}(\mathsf{ck}, m_b) & \\ b' \leftarrow\!\$\ \mathcal{A}(\mathsf{ck}, \mathsf{cm}) & \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Definition 3.6 (Binding). *A commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is binding if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ (\mathsf{cm}, m_0, m_1, r_0, r_1) \leftarrow\!\$\ \mathcal{A}(\mathsf{ck})\end{array} : \begin{array}{l} m_0 \neq m_1 \wedge \\ \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, m_0, r_0) = 1 \wedge \\ \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, m_1, r_1) = 1\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Dual-Group Construction for Vector of Messages We instantiate a rerandomizable commitment scheme to a vector of messages in the bilinear group setting as per the construction in [TBA+22] to enable efficient integration with our signature scheme. Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order p with an efficient Type-3 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

- $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{ck}$: Sample generators $(g, \tilde{g}) \leftarrow\!\$\ \mathbb{G}_1 \times \mathbb{G}_2$. For $i \in [1, n]$: Sample $y_i \leftarrow\!\$\ \mathbb{Z}_p$, compute $(g_i, \tilde{g}_i) \leftarrow (g^{y_i}, \tilde{g}^{y_i})$. Return $\mathsf{ck} \leftarrow (g, (g_1, \ldots, g_n), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_n))$.
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{attrs}) \to (\mathsf{cm}, \widetilde{\mathsf{cm}}, r)$: Parse ck as $(g, (g_1, \ldots, g_n), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_n))$. Sample $r \leftarrow\!\$\ \mathbb{Z}_p$. Compute $\mathsf{cm} \leftarrow g^r \prod_{i=1}^n g_i^{m_i}$ and $\widetilde{\mathsf{cm}} \leftarrow \tilde{g}^r \prod_{i=1}^n \tilde{g}_i^{m_i}$. Return $(\mathsf{cm}, \widetilde{\mathsf{cm}}, r)$.
- $\mathsf{Open}(\mathsf{ck}, (\mathsf{cm}, \widetilde{\mathsf{cm}}), \mathsf{attrs}, r) \to \{0, 1\}$: Check if $\mathsf{cm} = g^r \prod_{i=1}^n g_i^{m_i}$ and $\widetilde{\mathsf{cm}} = \tilde{g}^r \prod_{i=1}^n \tilde{g}_i^{m_i}$. Additionally, verify $e(\mathsf{cm}, \tilde{g}) = e(g, \widetilde{\mathsf{cm}})$. Return 1 if all checks pass, 0 otherwise.
- $\mathsf{Rerand}(\mathsf{ck}, (\mathsf{cm}, \widetilde{\mathsf{cm}}), \Delta_r) \to (\mathsf{cm}', \widetilde{\mathsf{cm}}', r')$: Compute $\mathsf{cm}' \leftarrow \mathsf{cm} \cdot g^{\Delta_r}$ and $\widetilde{\mathsf{cm}}' \leftarrow \widetilde{\mathsf{cm}} \cdot \tilde{g}^{\Delta_r}$. Set $r' \leftarrow r + \Delta_r$. Return $(\mathsf{cm}', \widetilde{\mathsf{cm}}', r')$.

This construction preserves the message vector while updating the randomness, making rerandomized commitments computationally indistinguishable from fresh commitments to the same message.

3.1.3 Rerandomizable Signature over Commitments

Definition 3.7 (Signature Scheme). *A signature scheme* Sig *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *where:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$*: is a probabilistic algorithm that takes as input a security parameter* λ *and outputs a secret signing key* sk *and a public verification key* pk*. The message space* $\mathcal{M}$ *is implicitly defined by* pk*.*
- $\mathsf{Sign}(\mathsf{sk}, m; r) \to \sigma$*: is a probabilistic algorithm that takes as input the secret key* sk*, a message* $m \in \mathcal{M}$*, and random coins* r *sampled from the randomness space* $\mathcal{R}$*. It outputs a signature* σ*.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to b$*: is a deterministic algorithm that takes as input the public key* pk*, a message* $m \in \mathcal{M}$*, and a signature* σ*. It outputs a bit* $b \in \{0, 1\}$*, where 1 indicates acceptance and 0 indicates rejection.*

Definition 3.8 (Correctness). *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is correct if for all* $k \in \mathbb{N}$*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{KeyGen}(1^k)]$ *and all* $m \in \mathcal{M}$ *we have:*

$$\Pr[\mathsf{Verify}(m, \mathsf{Sign}(m, \mathsf{sk}), \mathsf{pk}) = 1] = 1.$$

Definition 3.9 (EUF-CMA Security). *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is existentially unforgeable under adaptive chosen-message attacks if for all PPT algorithms* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow\!\$\ \mathcal{A}^{\mathcal{O}_{\mathsf{sk}}}(\mathsf{pk})\end{array} : \begin{array}{l} m^* \notin Q \wedge \\ \mathsf{Verify}(m^*, \sigma^*, \mathsf{pk}) = 1\end{array}\right] \leq \mathsf{negl}$$

where Q *is the set of queries made to* $\mathcal{O}_{\mathsf{sk}}$ *with access to* sk *is defined by:*

$$\textit{Oracle } \mathcal{O}_{\mathsf{sk}}(m): \ \textit{Returns } \sigma \leftarrow \mathsf{Sign}(m, \mathsf{sk})$$

Definition 3.10 (Rerandomizable Signature). *A rerandomizable signature scheme over commitments* RS *extends a standard signature scheme with the following interface:*

- $\mathsf{KeyGen}(1^\lambda, \mathsf{ck}) \to (\mathsf{sk}, \mathsf{pk})$*: Takes security parameter* λ *and commitment key* ck*, outputs signing key* sk *and verification key* pk*.*
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{cm}; u) \to \sigma$*: Signs commitment* cm *using signing key* sk *and randomness* u*.*
- $\mathsf{Rerand}(\mathsf{pk}, \sigma, \Delta_r, \Delta_u) \to \sigma'$*: Creates a rerandomized signature* σ' *from signature* σ *using randomization values* Δ_r, Δ_u*.*
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{cm}, \sigma) \to \{0,1\}$*: Verifies signature* σ *on commitment* cm *using public key* pk*.*

Definition 3.11 (Correctness). *A rerandomizable signature scheme over commitments satisfies correctness if for all security parameters* λ*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{KeyGen}(1^\lambda, \mathsf{ck})]$*, all messages* $m \in \mathcal{M}$*, all valid commitments* $\mathsf{cm} = \mathsf{CM.Commit}(\mathsf{ck}, m, r)$*, and all randomness values* u, Δ_r, Δ_u*:*

1. ***Basic Verification****:* $\mathsf{Verify}(\mathsf{pk}, \mathsf{cm}, \mathsf{Sign}(\mathsf{sk}, \mathsf{cm}; u)) = 1$
2. ***Rerandomization Consistency****:* $\mathsf{Verify}(\mathsf{pk}, \mathsf{CM.Rerand}(\mathsf{ck}, \mathsf{cm}, \Delta_r), \mathsf{Rerand}(\mathsf{pk}, \sigma, \Delta_r, \Delta_u)) = 1$ *where* $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{cm}; u)$
3. ***Rerandomization Equivalence****: The distribution of* $\mathsf{Rerand}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, \mathsf{cm}; u), \Delta_r, \Delta_u)$ *is computationally indistinguishable from* $\mathsf{Sign}(\mathsf{sk}, \mathsf{CM.Rerand}(\mathsf{ck}, \mathsf{cm}, \Delta_r); u + \Delta_u)$

Definition 3.12 (EUF-CMA). *A rerandomizable signature scheme over commitments is existentially unforgeable under adaptive chosen message (commitment) attacks if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{ll} \mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda), & \\ \mathsf{ck} \leftarrow \mathsf{CM.KeyGen}(\mathsf{BG}), & \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{BG}), & : \quad \mathsf{cm}^* = \mathsf{CM.Com}(\mathsf{ck}, m^*, r^*) \wedge \\ (m^*, \mathsf{cm}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) & \quad \mathsf{RS.Ver}(\mathsf{pk}, \sigma^*, m^*) = 1 \wedge \\ & \quad \mathsf{cm}^* \notin Q_{\mathsf{cm}} \end{array}\right] \leq \mathsf{negl}$$

where Q_{cm} *is the set of all commitments queried to the signing oracle*

3.1.4 Zero Knowledge Proofs and Sigma-protocols

Proof Relations

Definition 3.13 (Relation). *Let* $\mathcal{R} = \{((x)(w), crs)\} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ *be a binary relation between an input string* x*, a witness string* w*, and a common reference string* crs*. We define* $L_R = \{x : \exists w \text{ such that } (x, w, crs) \in R\}$*.*

Interactive Zero-Knowledge Proofs

Definition 3.14 (Interactive Zero-Knowledge Proof). *An interactive zero-knowledge protocol describes a system between a prover* P *and a verifier* V *that satisfies the following properties:*

- ***Completeness****: If* $(x, w, crs) \in \mathcal{R}$*, then* $\Pr[(P(w), V)(x, crs) = 1] \geq 1 - \mathsf{negl}(\lambda)$*.*
- ***Soundness with Knowledge Extraction****: For any prover* P^* *that convinces* V *with probability* $p(x) > \kappa(x)$ *(where* κ *is the knowledge error function), there exists a probabilistic polynomial-time knowledge extractor* $\mathcal{E}$ *such that given oracle access to* P^**,* $\mathcal{E}$ *outputs* w *satisfying* $(w, x, crs) \in R$ *within expected time proportional to* $\frac{|x|^c}{p(x) - \kappa(x)}$*.*

– ***Honest Verifier Zero-Knowledge****: There exists a simulator $\mathcal{S}$ such that for all $(x, w, crs) \in R$ and any (possibly malicious) verifier V^*, the view of V^* in the real interaction with $P(w)$ is computationally indistinguishable from $\mathcal{S}(x, crs)$.*

Remark 3.15 (Extractor). A proof is a proof of knowledge if an extractor $\mathcal{E}$, with rewind access to $\mathcal{P}^*$, can extract w when $\mathcal{P}^*$ convinces $\mathcal{V}$ with non-negligible probability:

$$\Pr[\mathcal{E}^{\mathcal{P}^*}(x) = w : (x, w) \in R] \geq \Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] - \mathsf{negl}(\lambda).$$

Remark 3.16 (Zero Knowledge Simulator). A proof is zero-knowledge if it reveals only that $x \in L$. Formally, for any $\mathcal{V}^*$, there exists a simulator $\mathcal{S}$ such that:

$$\{\text{VIEW}_{\mathcal{V}^*}(\mathcal{P}(w), \mathcal{V}^*)(x)\} \approx_c \{\mathcal{S}(x)\}, \quad \forall x \in L.$$

Sigma-Protocols

Definition 3.17 (Sigma Protocol). *A Sigma-protocol for a relation R is a three-move protocol:*

1. *$\mathcal{P}(x, w)$ sends commitment a.*
2. *$\mathcal{V}$ sends challenge $e \leftarrow \{0,1\}^t$.*
3. *$\mathcal{P}$ sends response z.*

$\mathcal{V}$ accepts if $\phi(x, a, e, z) = 1$. It satisfies completeness, special soundness, and SHVZK.

Schnorr Sigma Protocol For $\mathcal{R}_{\mathsf{DL}} = \{(h, w) \in \mathbb{G} \times \mathbb{Z}_q : h = g^w\}$, Schnorr's protocol proves knowledge of w:

– **Commitment**: $\mathcal{P}$ samples $r \leftarrow \mathbb{Z}_q$, sends $a = g^r$.

– **Challenge**: $\mathcal{V}$ sends $e \leftarrow \{0,1\}^t$.

– **Response**: $\mathcal{P}$ sends $z = r + ew \mod q$.

– **Verification**: $\mathcal{V}$ checks $g^z = a \cdot h^e$.

Properties:

– **Completeness**: $g^z = g^{r+ew} = g^r \cdot (g^w)^e = a \cdot h^e$.

– **Special Soundness**: From (a, e, z) and (a, e', z'), $w = (z - z')/(e - e') \mod q$.

– **SHVZK**: Simulator samples $z \leftarrow \mathbb{Z}_q$, sets $a = g^z h^{-e}$.

3.2 Formal Treatment for UTT's Rerandomizable Signature Over Commitments

We build upon the rerandomizable signature scheme over commitments introduced in UTT [TBA+22]. This signature scheme enables our anonymous credential system to present signatures that are both unlinkable and unforgeable without revealing the underlying identity attributes. Our main contribution is the first complete and tight security proof in the Algebraic Group Model that establishes EUF-CMA security with minimal assumptions (we note that [TBA+22] presents a proof sketch).

3.2.1 Position Binding Pedersen Commitments for a Vector of Messages

Building on the standard commitment scheme defined in Section 3.3, we extend it with the following properties:

Definition 3.18 (Position Binding). *A commitment scheme with message vectors in $\mathcal{M}^n$ is position binding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{l|l} & \mathsf{Open}(\mathsf{ck},\mathsf{cm},\mathsf{attrs}_0,r_0)=1 \wedge \\ \mathsf{ck} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda,1^n) & \mathsf{Open}(\mathsf{ck},\mathsf{cm},\mathsf{attrs}_1,r_1)=1 \wedge \\ (\mathsf{cm},i,\mathsf{attrs}_0,\mathsf{attrs}_1,r_0,r_1) \leftarrow\!\$\ \mathcal{A}(\mathsf{ck}) & \mathsf{attrs}_0[i] \neq \mathsf{attrs}_1[i] \wedge \\ & \mathsf{attrs}_0[j]=\mathsf{attrs}_1[j]\ \forall j \neq i \end{array}\right] \leq \mathsf{negl}(\lambda)$$

This ensures that an adversary cannot open a commitment to two different values at any single position while keeping other positions constant.

Definition 3.19 (Rerandomizability). *A commitment scheme* $\mathsf{Com} = (\mathsf{Setup},\mathsf{Commit},\mathsf{Open})$ *is rerandomizable if it includes an additional algorithm* Rerand *such that:*

- $\mathsf{Rerand}(\mathsf{ck},\mathsf{cm},\Delta_r) \to (\mathsf{cm}',r')$*: is a probabilistic algorithm that takes as input the commitment key* ck*, a commitment* cm*, and additional randomness Δ_r. It outputs a new commitment* cm' *and updated opening value r'.*

  For all $\lambda \in \mathbb{N}$, all $\mathsf{ck} \in [\mathsf{Setup}(1^\lambda)]$, all $m \in \mathcal{M}$, all $(\mathsf{cm},r) \in [\mathsf{Commit}(\mathsf{ck},m)]$, and all Δ_r:

  $$\Pr[(\mathsf{cm}',r') \leftarrow\!\$\ \mathsf{Rerand}(\mathsf{ck},\mathsf{cm},\Delta_r) : \mathsf{Open}(\mathsf{ck},\mathsf{cm}',m,r')=1] = 1.$$

  Furthermore, the distribution of cm' *should be computationally indistinguishable from a fresh commitment to the same message.*

3.2.2 SDLP and Position Binding Security Analysis in the AGM

Our dual-group Pedersen commitment scheme satisfies the following properties:

- **Perfect Hiding:** For any $\mathsf{attrs} \in \mathbb{Z}_p^n$, $\mathsf{Commit}(\mathsf{ck},\mathsf{attrs})$ outputs $(\mathsf{cm},\widetilde{\mathsf{cm}},r)$ where $\mathsf{cm} = g^r \prod_{i=1}^n g_i^{m_i}$ and $\widetilde{\mathsf{cm}} = \tilde{g}^r \prod_{i=1}^n \tilde{g}_i^{m_i}$. Since $r \leftarrow\!\$\ \mathbb{Z}_p$ is uniform, cm and $\widetilde{\mathsf{cm}}$ are uniformly distributed in $\mathbb{G}_1$ and $\mathbb{G}_2$, revealing no information about attrs.
- **Position Binding:** Under the SDLP assumption in the AGM, the scheme is position binding. The reduction 3.20 embeds an SDLP instance at a random position, using the adversary's algebraic break to extract the solution with advantage scaled by $1/n$.
- **Rerandomizability:** The Rerand algorithm outputs $(\mathsf{cm}',\widetilde{\mathsf{cm}}',r') = (\mathsf{cm}\cdot g^{\Delta_r}, \widetilde{\mathsf{cm}}\cdot \tilde{g}^{\Delta_r}, r+\Delta_r)$, which is a valid commitment to the same attrs. Since $\Delta_r \leftarrow\!\$\ \mathbb{Z}_p$ is uniform, the distribution matches a fresh commitment, satisfying rerandomizability perfectly by construction.
- **Binding:** Position binding implies standard binding, as any break of standard binding (distinct $\mathsf{attrs}_0 \neq \mathsf{attrs}_1$) violates position binding at some position.

Theorem 3.20. *In the Algebraic Group Model, if the Symmetric Discrete Logarithm Problem (SDLP) is hard in the bilinear group* $\mathsf{BG} = (\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,e,p,g,\tilde{g})$*, then the commitment scheme satisfies position binding. For any algebraic PPT adversary $\mathcal{A}$ against position binding, there exists a PPT reduction $\mathcal{B}$ against SDLP such that:*

$$\mathsf{Adv}^{\mathsf{pos\text{-}bind}}_{\mathcal{A},\mathsf{Com}}(\lambda) \leq n \cdot \mathsf{Adv}^{\mathsf{SDLP}}_{\mathcal{B},BG}(\lambda),$$

where n is the message vector length.

Proof. We construct a reduction where an algorithm $\mathcal{B}$ solves the Symmetric Discrete Logarithm Problem (SDLP) by leveraging an algebraic adversary $\mathcal{A}$ that breaks the position binding property of the commitment scheme.

1. **Setup: SDLP Challenger $\mathcal{C}$.** An external SDLP challenger provides $\mathcal{B}$ with an SDLP instance $(g^x, \tilde{g}^x) \in \mathbb{G}_1 \times \mathbb{G}_2$, where g and $\tilde{g}$ are generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (respectively) of prime order p, and $x \leftarrow\!\!\$\, \mathbb{Z}_p$ is a random, unknown exponent that $\mathcal{B}$ aims to compute.

2. **Reduction Algorithm $\mathcal{B}$.** $\mathcal{B}$ takes the SDLP instance and constructs a commitment key ck to feed to $\mathcal{A}$ as follows:

   (a) **Choose a random position.** Sample $i^* \leftarrow\!\!\$\, [1, n]$ uniformly at random, where n is the number of positions in the commitment scheme. This i^* is $\mathcal{B}$'s guess for where $\mathcal{A}$ will break position binding.

   (b) **Construct the commitment key ck.**

   – For position i^*, embed the SDLP instance:

   $$(g_{i^*}, \tilde{g}_{i^*}) = (g^x, \tilde{g}^x).$$

   – For all other positions $j \neq i^*$: Sample $y_j \leftarrow\!\!\$\, \mathbb{Z}_p$ independently, set $(g_j, \tilde{g}_j) = (g^{y_j}, \tilde{g}^{y_j})$.

   – Output $\mathsf{ck} = (g, (g_1, \ldots, g_n), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_n))$.

   (c) **Invoke the adversary.** $\mathcal{B}$ Runs $\mathcal{A}$ with input ck:

   $$\mathcal{A}(\mathsf{ck}) \to (\mathsf{cm}, \widetilde{\mathsf{cm}}, i, \mathsf{attrs}_0, \mathsf{attrs}_1, r_0, r_1).$$

   Here, $\mathcal{A}$ outputs a position binding break at position i, meaning $\mathsf{attrs}_0[i] \neq \mathsf{attrs}_1[i]$, and $\mathsf{attrs}_0[j] = \mathsf{attrs}_1[j]$ for all $j \neq i$, with cm opening to both (attrs_0, r_0) and (attrs_1, r_1).

3. **Adversary $\mathcal{A}$.** Since $\mathcal{A}$ is algebraic, it provides explicit representations of its outputs:

   – For $\mathsf{cm} \in \mathbb{G}_1$:

   $$\mathsf{cm} = g^{\alpha} \prod_{k=1}^{n} g_k^{\beta_k}$$

   where $\alpha, \beta_1, \ldots, \beta_n \in \mathbb{Z}_p$ are known to $\mathcal{B}$.

   – For $\widetilde{\mathsf{cm}} \in \mathbb{G}_2$:

   $$\widetilde{\mathsf{cm}} = \tilde{g}^{\tilde{\alpha}} \prod_{k=1}^{n} \tilde{g}_k^{\tilde{\beta}_k}$$

   where $\tilde{\alpha}, \tilde{\beta}_1, \ldots, \tilde{\beta}_n \in \mathbb{Z}_p$ are known to $\mathcal{B}$.

   The opening conditions hold:

   – $\mathsf{cm} = g^{r_0} \prod_{k=1}^{n} g_k^{m_{0,k}} = g^{r_1} \prod_{k=1}^{n} g_k^{m_{1,k}}$

   – $\widetilde{\mathsf{cm}} = \tilde{g}^{r_0} \prod_{k=1}^{n} \tilde{g}_k^{m_{0,k}} = \tilde{g}^{r_1} \prod_{k=1}^{n} \tilde{g}_k^{m_{1,k}}$

   – $e(\mathsf{cm}, \tilde{g}) = e(g, \widetilde{\mathsf{cm}})$.

   Extraction by $\mathcal{B}$. $\mathcal{B}$ uses $\mathcal{A}$'s output to solve for x:

   (a) **Check the position.** If $i \neq i^*$, abort. If $i = i^*$, proceed.

(b) **Substitute generators and equate exponents.** Substitute into cm: $g_k = g^{y_k}$ for $k \neq i^*$, and $g_{i^*} = g^x$. From the opening condition:

$$\mathsf{cm} = g^{r_0} \prod_{k=1}^{n} g_k^{m_{0,k}} = g^{r_0} \cdot \prod_{k \neq i^*} g^{y_k m_{0,k}} \cdot g^{x m_{0,i^*}},$$

and similarly:

$$\mathsf{cm} = g^{r_1} \cdot \prod_{k \neq i^*} g^{y_k m_{1,k}} \cdot g^{x m_{1,i^*}}.$$

Equate the exponents of g:

$$r_0 + \sum_{k \neq i^*} y_k m_{0,k} + x m_{0,i^*} = r_1 + \sum_{k \neq i^*} y_k m_{1,k} + x m_{1,i^*}.$$

(c) **Simplify the equation.** Since $\mathsf{attrs}_0[j] = \mathsf{attrs}_1[j]$ for $j \neq i$, we have $m_{0,k} = m_{1,k}$ for all $k \neq i^*$. Cancel these terms:

$$r_0 + x m_{0,i^*} = r_1 + x m_{1,i^*}.$$

Rearrange:

$$x(m_{0,i^*} - m_{1,i^*}) = r_1 - r_0.$$

(d) **Solve for x.** Since $i = i^*$ and $\mathsf{attrs}_0[i^*] \neq \mathsf{attrs}_1[i^*]$, it follows that $m_{0,i^*} \neq m_{1,i^*}$. Thus:

$$x = \frac{r_1 - r_0}{m_{0,i^*} - m_{1,i^*}} \mod p.$$

Output x as the solution to the SDLP instance.

Success Probability. $\mathcal{B}$ succeeds when $i = i^*$, which occurs with probability $1/n$. If $\mathcal{A}$ breaks position binding with probability ϵ, then $\mathcal{B}$ solves SDLP with probability ϵ/n.

Pairing Check. The condition $e(\mathsf{cm}, \tilde{g}) = e(g, \widetilde{\mathsf{cm}})$ ensures consistency across $\mathbb{G}_1$ and $\mathbb{G}_2$, but $\mathcal{B}$ relies solely on the $\mathbb{G}_1$ equations for extraction.

3.2.3 EUF-CMA Rerandomizable Signature over Commitments

We assume the existence of a commitment key ck from $\mathsf{CM.Setup}$ as input into our rerandomizable signature scheme RS. We copy the algorithm below for convenience.

- $\mathsf{CM.Setup}(1^\lambda, \ell, (y_i, \ldots, y_\ell \in \mathbb{Z}_p^\ell)) \to \mathsf{ck}$: Sample $(g, \tilde{g}) \leftarrow\!\$\ \mathbb{G}_1 \times \mathbb{G}_2$, For $i \in [1, \ell]$: Compute $g_i = g^{y_i}$ and $\tilde{g}_i = \tilde{g}^{y_i}$. Return $\mathsf{ck} \leftarrow (g, (g_1, \ldots, g_\ell), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_\ell))$
- $\mathsf{RS.KeyGen}(1^\lambda, \mathsf{ck}) \to (\mathsf{sk}, \mathsf{vk})$: Retrieve $(g, \cdot, \tilde{g}, \cdot)$ from ck, Sample $x \leftarrow\!\$\ \mathbb{Z}_p$, Set $(\mathsf{sk}, \mathsf{vk}) \leftarrow (g^x, \tilde{g}^x)$, return $(\mathsf{sk}, \mathsf{vk})$)
- $\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{cm}; u) \to \sigma$: Let $h \leftarrow g^u$ Return $\sigma \leftarrow (h, (\mathsf{sk} \cdot \mathsf{cm})^u)$
- $\mathsf{RS.Rerand}(\sigma, \Delta_r, \Delta_u) \to \sigma'$: Parse σ as (σ_1, σ_2) Set $\sigma_1' \leftarrow \sigma_1^{\Delta_u}$ Set $\sigma_2' \leftarrow (\sigma_2 \cdot \sigma_1^{\Delta_r})^{\Delta_u}$ Return $\sigma' \leftarrow (\sigma_1', \sigma_2')$
- $\mathsf{RS.Ver}(\mathsf{vk}, \mathsf{cm}, \sigma) \to \{0, 1\}$: Parse σ as (σ_1, σ_2), The prover $\mathcal{P}$ runs a Proof of Knowledge protocol with the following relation

$$\mathcal{R} \leftarrow \mathsf{PoK}\{(m_1, \ldots, m_\ell, r + \Delta_r) :$$

$$e(\sigma_2', \tilde{g}) = e(\sigma_1', \mathsf{vk}) \cdot e(\sigma_1', \widetilde{\mathsf{cm}}') \quad \wedge \quad e(\mathsf{cm}', \tilde{g}) = e(g, \widetilde{\mathsf{cm}}') \quad \wedge \quad \mathsf{cm}' = g^{r+\Delta_r} \prod_{i=1}^{\ell} g_i^{m_i}\}$$

3.2.4 Security Analysis in the AGM

Theorem 3.21. *The rerandomizable signature scheme is correct.*

Proof. First we demonstrate the prover's rerandomized signature verifies with the verification key vk and the rerandomized commitment. Essentially, we need the following pairing to hold:

$$e(\sigma'_2, \tilde{g}) = e(\sigma'_1, \mathsf{vk} \cdot \widetilde{\mathsf{cm}'})$$

We manipulate the bilinearity properties of the pairing groups to verify this equation:

$$\begin{aligned}
e(\sigma'_2, \tilde{g}) &= e((\mathsf{sk} \cdot \mathsf{cm})^{u \cdot \Delta_u} \cdot h^{\Delta_u \cdot \Delta_r}, \tilde{g}) \\
&= e(h^{x \cdot \Delta_u} \cdot \mathsf{cm}^{u \cdot \Delta_u} \cdot h^{\Delta_u \cdot \Delta_r}, \tilde{g}) \\
&= e(h^{x \cdot \Delta_u}, \tilde{g}) \cdot e(\mathsf{cm}^{u \cdot \Delta_u}, \tilde{g}) \cdot e(h^{\Delta_u \cdot \Delta_r}, \tilde{g}) \\
&= e(h^{\Delta_u}, \tilde{g}^x) \cdot e(\mathsf{cm}, \tilde{g})^{u \cdot \Delta_u} \cdot e(h^{\Delta_u}, \tilde{g})^{\Delta_r} \\
&= e(\sigma'_1, \mathsf{vk}) \cdot e(g^{u \cdot \Delta_u}, \widetilde{\mathsf{cm}}) \cdot e(\sigma'_1, \tilde{g})^{\Delta_r} \\
&= e(\sigma'_1, \mathsf{vk}) \cdot e(\sigma'_1, \widetilde{\mathsf{cm}}) \cdot e(\sigma'_1, \tilde{g}^{\Delta_r}) \\
&= e(\sigma'_1, \mathsf{vk} \cdot \widetilde{\mathsf{cm}} \cdot \tilde{g}^{\Delta_r}) \\
&= e(\sigma'_1, \mathsf{vk} \cdot \widetilde{\mathsf{cm}'})
\end{aligned}$$

Secondly, we need to verify knowledge of messages within the commitment. The Prover used $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$ during verification, which would be the natural method for a sigma-style proof of knowledge protocol, proving knowledge of the attributes of the commitment with $\mathbb{G}_2$ bases. However, due to the properties of the symmetric bilinear commitment 3.1, we can prove the equality of $\mathsf{cm}' \in \mathbb{G}_1$ and $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$ to reduce $\mathbb{G}_2$ computation on both the prover and verifier during verification. Thus the prover computes the following to prove equality of commitments across groups:

$$e(\mathsf{cm}', \tilde{g}) = e(g, \widetilde{\mathsf{cm}'})$$

Then proves the opening of the commitment in zero knowledge:

$$\Pi^{\mathcal{R}_{\mathsf{com}}} \leftarrow \mathsf{ZKPoK}\{(m_1, \ldots, m_\ell, r + \Delta_r) : \mathsf{cm}' = g^{r+\Delta_r} \prod_{i=1}^{\ell} g_i^{m_i}\}$$

Theorem 3.22 (EUF-CMA Security). *Assume the PS-LRSW assumption holds and the Pedersen commitment is computationally binding. Then, in the Algebraic Group Model, our rerandomizable signature scheme is existentially unforgeable under adaptive chosen-message(commitment) attacks. For any algebraic PPT adversary $\mathcal{A}$, there exist PPT reductions $\mathcal{B}_0, \mathcal{B}_1$ such that:*

$$\mathcal{A}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{RS},\mathcal{A}}(\lambda) \leq \mathcal{A}^{\mathsf{PS\text{-}LRSW}}_{\mathcal{B}_0}(\lambda) + \mathcal{A}^{\mathsf{Binding}}_{\mathcal{B}_1}(\lambda) + \frac{q_v + q_s}{p},$$

where q_v (verification) and q_s (signing) are query counts.

Proof. We construct two reductions handling different forgery types. Let $\mathcal{A}$ be an adversary with advantage ϵ.

Reduction Strategy: Since we don't know in advance which type of forgery $\mathcal{A}$ will produce, we design two reductions:

- $\mathcal{B}_0$ handles forgeries with a new message combination
- $\mathcal{B}_1$ handles forgeries where the same message appears in different commitments

1. Setup Given a PS-LRSW challenge $(g, \tilde{g}, X = g^x, \tilde{X} = \tilde{g}^x, Y = g^y, \tilde{Y} = \tilde{g}^y)$, both reductions:

- Sample $\alpha_1, \alpha_2, \beta_1, \beta_2 \leftarrow\!\!\$\ \mathbb{Z}_p$
- Set commitment base elements $g_i = Y^{\alpha_i} g^{\beta_i}$ and $\tilde{g}_i = \tilde{Y}^{\alpha_i} \tilde{g}^{\beta_i}$ for $i \in \{1, 2\}$
- Set $\mathsf{pk} = (\tilde{X}, \mathsf{ck} = (g, g_1, g_2, \tilde{g}, \tilde{g}_1, \tilde{g}_2))$
- Send pk to $\mathcal{A}$

2. Oracle Simulation For a signing query on commitment $\mathsf{cm} = g_1^{m_1} g_2^{m_2} g^r$:

- $\mathcal{B}_0$ (PS-LRSW Reduction):
  1. Compute $m = \alpha_1 m_1 + \alpha_2 m_2$
  2. Query PS-LRSW oracle to get (h, h^{x+my})
  3. Return $\sigma = (h, h^{x+my} \cdot h^{\beta_1 m_1 + \beta_2 m_2 + r})$
- $\mathcal{B}_1$ (Binding Reduction):
  1. Sample $u \leftarrow\!\!\$\ \mathbb{Z}_p$
  2. Compute $\sigma = (g^u, (X \cdot \mathsf{cm})^u)$

Verification Oracle:

- Parse $\sigma = (\sigma_1, \sigma_2)$
- Check $e(\sigma_2, \tilde{g}) = e(\sigma_1, \tilde{X} \cdot \widetilde{\mathsf{cm}})$ where $\widetilde{\mathsf{cm}} = \tilde{g}_1^{m_1} \tilde{g}_2^{m_2} \tilde{g}^r$
- Use AGM to extract exponents from σ_1, σ_2 if needed

3. Forgery Analysis When $\mathcal{A}$ outputs a forgery $(m_1^*, m_2^*, r^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$:

Case 1: If $m^* = \alpha_1 m_1^* + \alpha_2 m_2^*$ was never queried:

- Since $\mathcal{A}$ is algebraic, $\mathcal{B}_0$ knows the representation of σ_1^* as g^u
- $\mathcal{B}_0$ can compute $(g^u, \sigma_2^* / (\sigma_1^*)^{\beta_1 m_1^* + \beta_2 m_2^* + r^*})$
- This equals $(g^u, (g^x \cdot g^{ym^*})^u)$, breaking PS-LRSW

Case 2: If m^* was queried but with different message components:

- There exists a previous query $(m_1, m_2) \neq (m_1^*, m_2^*)$ but $\alpha_1 m_1 + \alpha_2 m_2 = \alpha_1 m_1^* + \alpha_2 m_2^*$
- This gives $g_1^{m_1} g_2^{m_2} = g_1^{m_1^*} g_2^{m_2^*}$, breaking the binding property
- $\mathcal{B}_1$ can extract the discrete logarithm relations, solving the binding challenge

Intuition for Security Our dual reduction approach handles all possible forgery types. The key insight is embedding the PS-LRSW challenge in a structured way that ensures:

1. If the adversary forges a signature for a new linear combination of messages, we break PS-LRSW
2. If the adversary reuses a linear combination but with different individual messages, we break binding

We ensure a forgery must break one underlying assumption, the tight reduction only uses a factor related to the number of oracle queries.

3.3 Extending UTT's Construction

3.3.1 Verify Speedup Improvement

In the original UTT $\mathbb{G}_1$ variant, verification involves two bilinear pairings equality checks and a proof of knowledge of the commitment:

$$e\big(\sigma_2', \tilde{g}\big) = e\big(\sigma_1', \widetilde{\mathsf{vk} \cdot \mathsf{cm}'}\big) \quad \wedge \quad e\big(\mathsf{cm}', \tilde{g}\big) = e\big(g, \widetilde{\mathsf{cm}'}\big) \quad \wedge \quad \mathsf{ZK.Verify}(\pi, \mathsf{cm}') = 1$$

Without the second pairing equality, the user must compute the proof of knowledge on $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$, which comes with its own overhead. That's because the signature is verified with $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$.

By redefining the signature in $\widetilde{\sigma} = \big(\widetilde{\sigma}_1, \widetilde{\sigma}_2\big) \in \mathbb{G}_2$ we can compute the proof of knowledge in $\mathbb{G}_1$ directly, reducing one pairing. The overhead of verifying the signature in $\mathbb{G}_2$ is significantly better than the additional pairing yielding a 20%–40% decrease in verification cost (see Table 7.2.1).

Verification Procedure The optimized verification algorithm $\mathsf{Verify}(\widetilde{\mathsf{vk}}, \mathsf{cm}', \widetilde{\sigma}') \to \{0, 1\}$ comprises:

$$e\big(g, \widetilde{\sigma}_2'\big) = e\big(\mathsf{vk} \cdot \mathsf{cm}', \widetilde{\sigma}_1'\big) \quad \wedge \quad \mathsf{ZK.Verify}(\pi, \mathsf{cm}') = 1$$

Correctness follows by bilinearity:

1. Prove $e(g, \widetilde{\sigma_2}') = e(\mathsf{vk} \cdot \mathsf{cm}', \widetilde{\sigma_1}')$

$$\begin{aligned}
e(g, \widetilde{\sigma_2}') &= e(g, (\widetilde{\sigma_2} \cdot \widetilde{\sigma_1}^{r_\Delta})^{u_\Delta}) \\
&= e(g, (\widetilde{\mathsf{sk}} \cdot \widetilde{\mathsf{cm}}^{u})^{u_\Delta} \cdot \tilde{h}^{r_\Delta \cdot u_\Delta}) \\
&= e(g, \widetilde{\mathsf{sk}}^{u_\Delta}) \cdot e(g, \widetilde{\mathsf{cm}}^{u+u_\Delta}) \cdot e(g, \tilde{h}^{r_\Delta \cdot u_\Delta}) \\
&= e(g^x, \widetilde{h}^{u_\Delta}) \cdot e(g, \widetilde{\mathsf{cm}})^{u+u_\Delta} \cdot e(g, \tilde{h}^{u_\Delta})^{r_\Delta} \\
&= e(\mathsf{vk}, \widetilde{h}^{u_\Delta}) \cdot e(\mathsf{cm}, \widetilde{h}^{u_\Delta}) \cdot e(g^{r_\Delta}, \tilde{h}^{u_\Delta}) \\
&= e(\mathsf{vk} \cdot \mathsf{cm} \cdot g^{r_\Delta}, \sigma_1') \\
&= e(\mathsf{vk} \cdot \mathsf{cm}', \sigma_1')
\end{aligned}$$

2. then PoK

$$\pi \leftarrow \mathsf{PoK}\{(r + \Delta_r, m_1, \ldots, m_\ell) : \mathsf{cm}' = g^{r+\Delta_r} \prod_{i=1}^{\ell} g_i^{m_i}$$

Performance Trade–Offs Although the G2 variant reduces verification latency by eliminating one pairing, it increases issuance cost by a factor of 1.3×–1.7× (Table 7.2.1). This trade–off is justified because:

- Credentials are issued infrequently but verified repeatedly.
- Issuance occurs on high–capacity servers, while verification may run on resource–constrained devices.
- The absolute increase in issuance latency (1–2 ms) is negligible in typical workflows.

3.3.2 Malicious Issuer Security Guarnatees

Our first improvement to the PS-UTT rerandomizable signature scheme is the addition of algorithm $\mathsf{RS.VerKey}$, which addresses a vulnerability in traditional anonymous credential systems that assume honest issuers. This assumption leaves users exposed to privacy breaches and potential forgeries from malicious credential providers. Through $\mathsf{RS.VerKey}$, we require issuers to prove knowledge of their secret key $\mathsf{sk} = g^x$ and demonstrate the well-formedness of the commitment key $\mathsf{ck} = \{g_i = g^{y_i}, \tilde{g}_i = \tilde{g}^{y_i}\}_{i\in[\ell]}$ via a zero-knowledge Sigma-protocol. This prevents issuers from constructing keys with hidden mathematical relationships that could deanonymize users during credential presentations or enable credential forgery, ensuring that anonymity guarantees remain intact even against adversarial issuers who attempt to subvert the cryptographic foundations of the system.

– $\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) \to \{0, 1\}$: is an interactive algorithm run by the issuer and the user requesting a signature that verifies the correctness of the issuer's secret and verification key $(\mathsf{sk}, \mathsf{vk})$ and commitment key ck. Issuer parses ck as $\{g^{y_i}, \tilde{g}^{y_i}\}_{i\in[\ell]}$ and runs a Σ-protocol 3.3.2 to prove the correctness of vk and ck

Protocol 1: Proving Correctness of Issuer Keys

Common Input: verification key $\mathsf{vk} = \tilde{g}^x \in \mathbb{G}_2$, commitment key $\mathsf{ck} = (g_1, \ldots, g_\ell, g, \tilde{g}_1, \ldots, \tilde{g}_\ell, \tilde{g})$

Prover Input: $x, y_1, \ldots, y_\ell \in \mathbb{Z}_q$ such that $\mathsf{vk} = \tilde{g}^x$ and $\{\mathsf{ck}_i = g^{y_i}, \widetilde{\mathsf{ck}}_i = \tilde{g}^{y_i}\}_{i\in[\ell]}$

Relation:

$$\mathcal{R}_{\mathsf{verkey}} = \left\{ \begin{matrix}(\mathsf{vk}, \mathsf{ck}, \widetilde{\mathsf{ck}}),\\ (x, \{y_i\}_{i=1}^{\ell})\end{matrix} \;\middle|\; \mathsf{vk} = \tilde{g}^x \wedge \{\mathsf{ck}_i = g^{y_i}, \widetilde{\mathsf{ck}}_i = \tilde{g}^{y_i}\}_{i\in[\ell]} \right\}$$

1. **Commitment:** Prover samples $r_x \leftarrow\!\!\$\, \mathbb{Z}_q$, $r_1, \ldots, r_n \leftarrow\!\!\$\, \mathbb{Z}_q$ and computes:

$$T_x = g^{r_x}, \quad \widetilde{T_x} = \tilde{g}^{r_x}, \quad T_i = g^{r_i}, \quad \widetilde{T_i} = \tilde{g}^{r_i} \quad \text{for each } i$$

   Sends T_x, $\widetilde{T_x}$, $(T_i, \widetilde{T_i})_{i=1}^n$ to the verifier.

2. **Challenge:** Verifier samples $c \leftarrow\!\!\$\, \mathbb{Z}_q$ and sends c to the prover.

3. **Response:** Prover computes:

$$z_x = r_x + c \cdot x, \quad z_i = r_i + c \cdot y_i \quad \text{for each } i$$

   Sends z_x, $(z_i)_{i=1}^n$ to the verifier.

4. **Verification:** Verifier checks:

$$\tilde{g}^{z_x} \stackrel{?}{=} \widetilde{T_x} \cdot \mathsf{vk}^c \quad \wedge \quad \tilde{g}^{z_i} \stackrel{?}{=} \widetilde{T_i} \cdot \tilde{g}_i^c \quad \wedge \quad e(T_i, \tilde{g}) \stackrel{?}{=} e(g, \widetilde{T_i}) \text{ for each } i$$

Security Analysis We separate our security analysis of the verification key vk and commitment key ck for clarity:

Theorem 3.23. *The Σ-protocol for $\mathsf{RS.VerKey}$, defined in Protocol 3.3.2, is complete, sound, and zero-knowledge, proving knowledge of $x \in \mathbb{Z}_q$ such that $\mathsf{vk} = \tilde{g}^x \in \mathbb{G}_2$, where $\pi_{\mathsf{vk}} = (\widetilde{T_x}, c, z_x)$ is the proof transcript.*

Proof. – **Completeness:** For an honest prover who knows x such that $\mathsf{vk} = \tilde{g}^x$, the prover samples $r_x \leftarrow\!\!\$\, \mathbb{Z}_q$, computes $\widetilde{T_x} = \tilde{g}^{r_x}$, and, upon receiving challenge c, responds with $z_x = r_x + c \cdot x$. The

verifier checks:

$$\begin{aligned}\tilde{g}^{z_x} &\stackrel{?}{=} \widetilde{T_x} \cdot \mathsf{vk}^c \\ &\stackrel{?}{=} \tilde{g}^{r_x} \cdot \tilde{g}^{xc} \\ &\stackrel{?}{=} \tilde{g}^{r_x + xc} \\ &= \tilde{g}^{z_x} \checkmark\end{aligned}$$

This holds, proving completeness.

– **Soundness:** There exists a knowledge extractor $\mathcal{E}$ that, given two accepting transcripts $(\widetilde{T_x}, c, z_x)$ and $(\widetilde{T_x}, c', z_x')$ with $c \neq c'$, computes $x = \frac{z_x - z_x'}{c - c'}$. From the verification equations:

$$\tilde{g}^{z_x} = \widetilde{T_x} \cdot \mathsf{vk}^c \quad \text{and} \quad \tilde{g}^{z_x'} = \widetilde{T_x} \cdot \mathsf{vk}^{c'},$$

subtracting exponents (since $\widetilde{T_x}$ is the same in both transcripts) yields $\tilde{g}^{z_x - z_x'} = \mathsf{vk}^{c - c'}$, so $\mathsf{vk} = \tilde{g}^{\frac{z_x - z_x'}{c - c'}}$. Given $\mathsf{vk} = \tilde{g}^x$, it follows that $x = \frac{z_x - z_x'}{c - c'}$, proving soundness.

– **Zero-Knowledge:** The simulator $\mathcal{S}$, given challenge c, samples $z_x \leftarrow\!\!\$\ \mathbb{Z}_q$ and sets $\widetilde{T_x} = \tilde{g}^{z_x} \cdot \mathsf{vk}^{-c}$. This satisfies the verification equation:

$$\tilde{g}^{z_x} = \widetilde{T_x} \cdot \mathsf{vk}^c.$$

In the real protocol, $\widetilde{T_x} = \tilde{g}^{r_x}$ for random r_x, and $z_x = r_x + cx$, making $\widetilde{T_x} = \tilde{g}^{z_x - cx}$. Since z_x is uniform in $\mathbb{Z}_q$, $\widetilde{T_x}$ is uniform in $\mathbb{G}_2$, identical to the real protocol's distribution, proving honest-verifier zero-knowledge.

Theorem 3.24. *The Σ-protocol for* RS.VerKey*, defined in Protocol 3.3.2, is complete, sound, and zero-knowledge, proving knowledge of $\{y_i\}_{i \in [\ell]} \in \mathbb{Z}_q$ such that $(\mathsf{ck}_i, \widetilde{\mathsf{ck}}_i)$ both exponentiate the same y such that $(e(\mathsf{ck}_i, \widetilde{\mathsf{ck}}_i) = e(g^{y_i}, \tilde{g}^{y_i}) = e(g, \tilde{g}))$ for each $i \in [\ell]$*

Proof. – **Completeness:** For an honest prover who knows y_i such that ck_i and $\widetilde{\mathsf{ck}}_i$ commit to the same y_i. For each i, the prover samples $r_i \leftarrow\!\!\$\ \mathbb{Z}_q$, computes $T_i = g^{r_i}, \widetilde{T}_i = \tilde{g}^{r_i}$, receives challenge c, responds with $z_i = c \cdot y_i$. The verifier checks

$$\begin{aligned}g^{z_i} &\stackrel{?}{=} T_i \cdot \mathsf{ck}_i^c \\ &\stackrel{?}{=} g^{r_i} \cdot (g^{y_i})^c \\ &\stackrel{?}{=} g^{r_i} \cdot g^{y_i \cdot c} \\ &\stackrel{?}{=} g^{r_i + y_i \cdot c} \\ &= g^{z_i} \checkmark\end{aligned} \qquad \begin{aligned}\tilde{g}^{z_i} &\stackrel{?}{=} \widetilde{T}_i \cdot \widetilde{\mathsf{ck}}_i^{\,c} \\ &\stackrel{?}{=} \tilde{g}^{r_i} \cdot (\tilde{g}^{y_i})^c \\ &\stackrel{?}{=} \tilde{g}^{r_i} \cdot \tilde{g}^{y_i \cdot c} \\ &\stackrel{?}{=} \tilde{g}^{r_i + y_i \cdot c} \\ &= \tilde{g}^{z_i} \checkmark\end{aligned} \qquad \begin{aligned}e(T_i, \tilde{g}) &\stackrel{?}{=} e(g, \widetilde{T}_i) \\ &\stackrel{?}{=} e(g, \tilde{g}^{r_i}) \\ &\stackrel{?}{=} e(g^{r_i}, \tilde{g}) \\ &\stackrel{?}{=} e(T_i, \tilde{g}) \checkmark\end{aligned}$$

– **Soundness:** There exists a knowledge extractor $\mathcal{E}$ that, given two accepting transcripts $(T_i, \widetilde{T}_i, c, z_i)$ and $(T_i, \widetilde{T}_i, c', z_i')$ with $c \neq c'$, computes $y_i = \frac{z_i - z_i'}{c - c'}$. From the verification equations:

$$g^{z_i} = T_i \cdot \mathsf{ck}_i^c \quad \text{and} \quad g^{z_i'} = T_i \cdot \mathsf{ck}_i^{c'},$$

subtracting exponents (since T_i is the same in both transcripts) yields $g^{z_i - z_i'} = \mathsf{ck}_i^{c - c'}$, so $\mathsf{ck}_i = g^{\frac{z_i - z_i'}{c - c'}}$. Given $\mathsf{ck}_i = g^{y_i}$, it follows that $y_i = \frac{z_i - z_i'}{c - c'}$.

Similarly, from the G2 component verification:

$$\tilde{g}^{z_i} = \widetilde{T}_i \cdot \widetilde{\mathsf{ck}}_i^{\,c} \quad \text{and} \quad \tilde{g}^{z_i'} = \widetilde{T}_i \cdot \widetilde{\mathsf{ck}}_i^{\,c'},$$

we can derive $\widetilde{\mathsf{ck}}_i = \tilde{g}^{\frac{z_i - z_i'}{c - c'}}$, confirming that $\widetilde{\mathsf{ck}}_i = \tilde{g}^{y_i}$ with the same y_i.

The pairing check $e(T_i, \tilde{g}) = e(g, \widetilde{T}_i)$ ensures that $T_i = g^{r_i}$ and $\widetilde{T}_i = \tilde{g}^{r_i}$ use the same randomness r_i, which is crucial for the extractor to obtain consistent discrete logarithms from both group components. This consistency guarantees that the extracted y_i values from G1 and G2 equations are identical, proving soundness

– **Zero-Knowledge:** We need to construct a simulator $\mathcal{S}$ that, given challenge c, can produce a transcript indistinguishable from a real interaction without knowing the witnesses y_i. The simulator works as follows:

For each i, simulator $\mathcal{S}$ Samples $z_i \leftarrow\!\$\, \mathbb{Z}_q$, computes $T_i = g^{z_i} \cdot \mathsf{ck}_i^{-c}$ and $\widetilde{T}_i = \tilde{g}^{z_i} \cdot \widetilde{\mathsf{ck}}_i^{\,-c}$

The simulated transcript $(T_i, \widetilde{T}_i, c, z_i)$, which passes verification:

$$\begin{aligned} g^{z_i} &= T_i \cdot \mathsf{ck}_i^c \\ &= (g^{z_i} \cdot \mathsf{ck}_i^{-c}) \cdot \mathsf{ck}_i^c \\ &= g^{z_i} \checkmark \end{aligned} \qquad \begin{aligned} \tilde{g}^{z_i} &= \widetilde{T}_i \cdot \widetilde{\mathsf{ck}}_i^{\,c} \\ &= (\tilde{g}^{z_i} \cdot \widetilde{\mathsf{ck}}_i^{\,-c}) \cdot \widetilde{\mathsf{ck}}_i^{\,c} \\ &= \tilde{g}^{z_i} \checkmark \end{aligned}$$

For the pairing check, we need to show:

$$e(T_i, \tilde{g}) = e(g, \widetilde{T}_i)$$
$$e(g^{z_i} \cdot \mathsf{ck}_i^{-c}, \tilde{g}) = e(g, \tilde{g}^{z_i} \cdot \widetilde{\mathsf{ck}}_i^{\,-c})$$

This equality holds because:

$$\begin{aligned} e(g^{z_i} \cdot \mathsf{ck}_i^{-c}, \tilde{g}) &= e(g^{z_i}, \tilde{g}) \cdot e(\mathsf{ck}_i^{-c}, \tilde{g}) \\ &= e(g, \tilde{g}^{z_i}) \cdot e(g^{y_i \cdot (-c)}, \tilde{g}) \\ &= e(g, \tilde{g}^{z_i}) \cdot e(g, \tilde{g}^{y_i \cdot (-c)}) \\ &= e(g, \tilde{g}^{z_i} \cdot \tilde{g}^{y_i \cdot (-c)}) \\ &= e(g, \tilde{g}^{z_i} \cdot \widetilde{\mathsf{ck}}_i^{\,-c}) \\ &= e(g, \widetilde{T}_i) \checkmark \end{aligned}$$

The distribution of the simulated transcript is identical to that of a real transcript: in both cases, z_i is uniformly distributed in $\mathbb{Z}_q$, which makes T_i and $\widetilde{T}_i$ uniform elements in their respective groups. Therefore, the protocol is zero-knowledge.

3.4 Expressive Proof Evaluation

Proof System	**Expressiveness**	**Efficiency (Proof Size)**	**Efficiency (Verification)**	**Trusted Setup**	**Key Assumptions**
Sigma Protocols	Linear relations, AND/OR	$O(\lambda \cdot k)$ bits	$O(k)$ exponentiations	No	Discrete Log
Groth-Sahai	Quadratic relations	$O(1)$ group elements	$O(n)$ pairings	Yes (CRS)	Pairing-based
zk-SNARKs	General circuits	∼200 bytes	$O(1)$ pairings	Yes	QAP/RICS

λ: security parameter; k: number of statements; n: relation size.

Anonymous Credentials requires users to prove knowledge of secrets (e.g. attributes like age, citizenship) without leaking additional information. The "Anonymous" term comes from this fact, the user can remain anonymous while using a credential. Using must mean it is useful.

Proof System Choice and Credential Structure The choice of anonymous credential (algebraic structure) will determine which proof system can be used The algebraic structure of an anonymous credential will limit which proof system that can be used by a credential. The proof system will determine what kind of statements can be proven in zero knowledge. The choice of proof system will determine the efficiency and expressiveness of the proofs.

Three categories of expressiveness: 1. Linear relations: Sigma protocols are very efficient and can prove linear relations

2. Quadratic Relations: Groth Sahai proofs can prove quadratic relations

3. General Circuits: zk-SNARKS can

Expressiveness - Sigma Protocols: Prove linear relations (e.g., knowledge of discrete logarithms) and support composition (AND/OR proofs) but also Equality, Bit Composition, Range Proof, Set Membership.

- Groth-Sahai: Handle quadratic relations in bilinear groups, ideal for pairing-based statements.

- zk-SNARKs: Support general arithmetic circuits, covering any NP statement.

– **Schnorr (Sigma Protocols):** These prove statements about discrete logarithms (e.g., "I know x such that $g^x = h$"). They're limited to linear relations but can be combined (e.g., proving "A or B" or "A and B"). In your work, this supports selective disclosure and attribute equality efficiently, as seen in PS-UTT22's sublinear scaling with multi-scalar multiplication (Page 92).

– **Groth-Sahai:** These extend to quadratic relations in bilinear groups (e.g., pairing equations). They're more expressive for pairing-based cryptography but less so than zk-SNARKs, which handle circuits. They're less efficient than sigma protocols for simple statements due to pairing costs (e.g., $O(n)$ pairings vs. $O(k)$ exponentiations).

Use-Case	**ZKP Requirement**	
Prove Valid Credential	Signature of Knowledge	Schnorr
Age > 18 or Cred NOT Expired	Greater/Less Than	Sigma with Range Proof (Bulletproof
Prove Attribute in a List (Set Membership)	Set Membership	
Prove Equality of Attributes	Equality of Attributes	
Prove Credential Not Revoked	Accumulator Non-Membership Proof	
Selective Disclosure of Attributes	Prove specific attributes while hiding others	

Table 1. Use-Cases and Corresponding Zero-Knowledge Proof Requirements

[1]This involves an inequality (birthdate < today - 18), which can be expressed as a linear relation using range proofs. No pairings or zk-SNARKs needed.

Simple Identity Use-cases - show, etc

More Complex (Piloted and tested) - organisation credentials / education - prescription, insurance, health - visas, passport, travel documents

Sigma Protocols - Motivation -

- Limitations (why we would need to use GS proofs)

Groth Sahai - Motivation

- Limitations (why we would need to use zk-SNARKS)

zk-SNARKS - Motivation

bulletproof? Combination

Motivate the use-case - simple possession (I have a valid credential) - I'm over 18 - My license is not expired - My country is in the list - My id is equal to this other id

Sigma Protocols - great for proving simple, linear relationships, like proving you know a secret number (a discrete logarithm) or that a secret number equals another secret number. - efficient, don't need trusted setup

Sigma protocols can't efficiently handle equations with a product of unknowns e.g. proving a pairing of 2 values satisfies a condition (Why is this important, why does this give more functionality)?

Groth Sahai - Pairing-based are needed for proving quadratic relations, common in set membership or non-membership proofs

zk-SNARKS - General arithmetic circuits that can prove the output of a hash function or a multi-step policy

Pairing-Based Proofs (e.g., Groth-Sahai): These excel at quadratic relations, common in set membership or non-membership proofs.

zk-SNARKs: These can prove almost anything (arbitrary computations), but they're heavier and require a trusted setup.

«< INCLUDE TABLE WITH SCHNORR PROOF COST »> «< INCLUDE METRICS FOR RANGE PROOF, REVOCATION LIST, ARITHMETIC RELATION, BOOLEAN RELATION »> «< INCLUDE COMPARISON BETWEEN SCHNORR BASED SIGNATURES, SPS-EQ and their proof functionality (as in improved anonymous credentials) and zk-SNARK based credentials»

- This says there's a major drawback but I prove it's not actually a drawback, it's only theoretical and it doesn't hold in practice.

A major drawback from such an approach is that the zero-knowledge proof used during showings is of variable-length and may require multiple sub-proofs On the other hand, more recent constructions (e.g., [11,14,24,32,38,46,49]) apply other techniques based on different lines of work to adapt the signature and the message without using Zero-Knowledge Proofs of Knowledge (ZKPoK), providing constant-size showings.

Many schemes spruik the O(1) verification, but is it as efficient as ours? Their computational complexity involves

SPS-EQ style

- Efficient Membership/Non-Membership -

Expressiveness

Sigma Protocols Range Proof Bit Decomposition Equality AND, OR, NAND, NOT

Expressiveness in proof systems $\Sigma-$protocols $\subset$ Quadratic GS Proofs $\subset$ Circuit SNARKS

Expressiveness in proof systems refers to the class of statements or relations you can prove without revealing witnesses.

Predicate Satisfaction: For credentials with attributes that satisfy a policy ϕ:

$$\mathcal{R}_\phi = \{(\{\mathsf{cm}_j\}, (\{\mathsf{attrs}_j\})) \mid \forall j : \mathsf{cm}_j \text{ correctly commits to } \mathsf{attrs}_j \wedge \phi(\{\mathsf{attrs}_j\}) = 1\}$$

A predicate ϕ is simply a policy statement about attributes that is satisfiable by the supported algebraic constraints in Sigma protocols, noting that sigma protocols are extremely expressive and can . For example:

These can be composed to support complex policies while maintaining zero-knowledge properties, revealing nothing beyond the fact that the policy is satisfied.

– "age $>$ 18" (for a single credential)

– "has valid driver's license AND passport" (requiring multiple credentials)

– "degree = 'Computer Science' AND university in accredited_list"

The Σ-protocol allows proving these statements are true without revealing the actual attribute values, only that they satisfy the predicate ϕ.

The Σ-protocol framework allows for highly expressive predicates ϕ including:

– **Boolean operations:** AND, OR, and NOT compositions of simpler predicates

– **Arithmetic relations:** Equality, inequality ($<$, $>$, $\leq$, $\geq$), and linear combinations

– **Range proofs:** Demonstrating a value lies within a specific range (e.g., "$18 \leq$ age < 65")

– **Set membership:** Proving an attribute belongs to an approved set without revealing which one

Predicate Satisfaction We define a predicate ϕ as a boolean function over an attribute vector attrs, formally $\phi : (\mathcal{A}) \to \{0, 1\}$, where $\mathcal{A}$ is the space of the attribute vectors. For a credential with attributes $\mathsf{attrs} = [\mathsf{id}, \mathsf{s}, \mathsf{ctx}, \mathsf{exp}]$, we say that "$\mathsf{attrs}$ satisfies ϕ", denoted as $\phi(\mathsf{attrs}) = 1$, if the boolean function evaluates to true on the attributes. For example, the predicate $\phi_{master} : \mathsf{ctx} = "master"$ is satisfied by $\mathsf{attrs} : [\mathsf{id} = "123", \mathsf{ctx} = "master"]$. Our system supports complex predicates such as $\phi : age > 18 \wedge country = US$ enabling expressive policies beyond simple equality checks. In our unforgeability definition, predicate satisfaction ensures an adversary cannot forge a proof for a predicate they do not legitimately satisfy beyond legitimately reusing existing credentials.

To demonstrate the practical impact of our optimizations, we compared our approach against alternative systems for the common use case of verifying credential expiration. Table 15 presents the results for generating and verifying a proof that a credential has not expired.

Table 2. Performance Comparison for Credential Operations

Approach	Scheme	Show (ms)	Verify (ms)	Proof Size
Simple Possession	[RWGM22]	2	2	424B
	Us	2	2	
Expiry	[RWGM22]	2	2	424B
	Us	2	2	
Linkable Show	[RWGM22]	41		
	Us	2	2	
Rate Limiting	[RWGM22]	58		
	Us	2	2	

Our evaluation reveals that our system can verify a credential's expiry status in just 3.3ms (combined Show+Verify), compared to 45-55ms for ZK-Creds approaches—a performance improvement of over 13×. This dramatic difference highlights how our optimized signature scheme and sigma protocol enables efficient predicate verification without sacrificing privacy.

Importantly, while general-purpose zero-knowledge systems provide flexibility, they introduce substantial computational overhead that makes interactive verification scenarios impractical. Our approach achieves similar expressiveness with dramatically better performance for the most common credential verification operations.

3.5 Expressive ABC from Our Construction

We construct an Attribute-Based Anonymous Credential System from our Rerandomizable Commitment and Signature schemes and prove its security in the model [FHS19]; we extend it to support predicate-based zero-knowledge proof verification, allowing users to prove statements about their committed and signed attributes without revealing any additional information.

3.5.1 Syntax

Definition 3.25 (ABC System). *An Attribute-based Anonymous Credential system consists of the following probabilistic polynomial-time (PPT) algorithms:*

- $\mathsf{Setup}(\lambda) \to (\mathsf{ppar})$*: Takes a security parameter* λ *and outputs public parameters* ppar*.*
- $\mathsf{OrgKeygen}(\mathsf{ppar}, \ell) \to (\mathsf{osk}, \mathsf{opk})$*: Takes public parameters* ppar *and an upper bound* ℓ *on the number of credential attributes, outputting an organization's secret key* osk *and public key* opk*.*
- $(\mathsf{Obtain}(\mathsf{attrs}, \mathsf{opk}), \mathsf{Issue}(\mathsf{osk}, \mathsf{aux})) \to (\mathsf{cred}, \perp)$*: An interactive protocol where the user inputs attributes* attrs *and the organization's public key* opk*, and the issuer inputs its secret key* osk *and auxiliary information* aux*. Outputs a credential* cred *to the user and* $\perp$ *to the issuer.*
- $(\mathsf{Show}(\mathsf{cred}, \mathsf{usk}, \phi), \mathsf{Verify}(\mathsf{cred}', \phi)) \to 0, 1$*: An interactive protocol where the user inputs a credential* cred*, a secret key* usk*, and a predicate* ϕ*, and the verifier inputs a credential presentation* cred' *and the predicate* ϕ*. Outputs 1 if the presentation satisfies the predicate, 0 otherwise.*

3.5.2 Security Model

Our Multi-Show Attribute-Based Anonymous Credential is defined by three properties:

- **Correctness:** When all parties follow the protocol honestly, a user with valid credentials should always be able to generate proofs for predicates satisfied by their attributes, which verifiers will accept with overwhelming probability.
- **Unforgeability:** No probabilistic polynomial-time adversary should be able to produce a valid proof for a predicate that they cannot legitimately satisfy based on credentials they have been issued.
- **Anonymity:** Proofs should reveal only that the predicate is satisfied, without identifying which specific user produced the proof, even if the verifier and issuer collude.

To model the adversary's capabilities and the system's state, we introduce the following lists and oracles:

Lists

- HU: The set of honest users whose secret keys remain unknown to the adversary $\mathcal{A}$
- CU: The set of corrupt users whose secret keys are known to the adversary $\mathcal{A}$
- CRED: A list tracking all issued credentials, where each credential is associated with a user and their attributes
- OWNR: A mapping from each credential to its owning user, i.e., $\mathsf{OWNR}[\mathsf{cred}] = i$ if credential cred belongs to user i

– SHOW: A list tracking all credential show outputs

Oracles

– $\mathcal{O}_{\mathsf{HU}}()$: Creates a new honest user i, adds them to HU, and returns i

– $\mathcal{O}_{\mathsf{CU}}(i)$: Corrupts user i by moving them from HU to CU, revealing their secret keys and all credentials owned by i

– $\mathcal{O}_{\mathsf{Obtain}}(i, \vec{m})$: Issues a credential cred to user i for the attribute vector $\vec{m}$, provided $i \in \mathsf{HU}$. The credential is added to CRED, and $\mathsf{OWNR}[\mathsf{cred}]$ is set to i

– $\mathcal{O}_{\mathsf{Show}}(i, \phi)$: Generates a proof π that the credentials of user i satisfy the predicate ϕ, provided $i \in \mathsf{HU}$ and the credentials meet the condition ϕ

Definition 3.26 (Correctness).

$$\Pr\left[\mathsf{Verify}(\mathsf{cred}', \mathsf{cm}', \phi, \pi) = 1 \mid \textit{all steps honest} \wedge \phi(m) = 1\right] = 1 - \mathsf{negl}(1^\lambda)$$

Intuition: An ABC system is correct if, when all parties follow the protocol honestly, a user can successfully prove a true statement about their credential to a verifier. Specifically, for all honestly generated public parameters, keys, credentials, and predicates satisfied by the user's attributes, the verification process accepts the proof with overwhelming probability.

$\mathrm{Game}^{\mathsf{UNF}}_{\mathsf{ABC},\mathcal{A}}(1^\lambda)$	$\mathrm{Game}^{\mathsf{ANON}}_{\mathsf{ABC},\mathcal{A}}(\lambda)$
// Challenger Setup	$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{HU} \leftarrow \emptyset, \mathsf{CU} \leftarrow \emptyset$ // Challenger Setup
Initialize $\mathsf{HU} \leftarrow \emptyset, \mathsf{CU} \leftarrow \emptyset,$	$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathcal{A}(\mathsf{OrgKeyGen}(\mathsf{pp}))$ // Adversary may corrupt issuer
$\mathsf{CRED} \leftarrow \emptyset, \mathsf{OWNR} \leftarrow \{\}$	For $i \in \{0,1\}$: // Setup two challenge users
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$	$\mathsf{r}_i \leftarrow \mathsf{UserKeyGen}(\mathsf{pp}), \mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
// $\mathcal{A}$ queries oracles	$\vec{m}_i \leftarrow \mathcal{A}(i)$ such that $\phi(\vec{m}_i) = 1$
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk})$	$\mathsf{cm}_i \leftarrow \mathsf{CM.Com}(\vec{m}_i; \mathsf{r}_i), \mathsf{cred}_i \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm}_i)$
// Forgery	$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk})$ // Learning Phase
$\mathcal{A}$ outputs $(\mathsf{cred}'^* = (\sigma'^*, \mathsf{cm}'^*), \phi^*, \pi^*)$	$\phi \leftarrow \mathcal{A}()$ // Challenge Phase
// Winning Condition	Assert $\phi(\vec{m}_0) = \phi(\vec{m}_1) = 1$ and $\{0,1\} \subset \mathsf{HU}$
$\mathsf{Verify}(\mathsf{cred}'^*, \phi^*, \pi^*, \mathsf{opk}) = 1 \wedge$	$b \leftarrow\!\!\$\ \{0,1\}$ // Challenger samples random bit
$\mathsf{OWNR}[\mathsf{cred}'^*] \notin \mathsf{CU} \ \vee \ \phi^*(\vec{m}^*) = 0$	$(\mathsf{cred}', \mathsf{cm}', \pi) \leftarrow \mathsf{Show}(\mathsf{cred}_b, \mathsf{cm}_b, \mathsf{r}_b, \phi)$
// i.e., either the credential belongs to an honest user	$b' \leftarrow \mathcal{A}(\mathsf{cred}', \mathsf{cm}', \pi)$ // Adversary guesses
// or does not satisfy the claimed predicate	**return** b', // Return the adversary guess

Fig. 1. Unforgeability and Anonymity Games for the ABC System

Definition 3.27 (Unforgeability). *An ABC system is unforgeable if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$\mathsf{Adv}\left[\mathrm{Game}^{\mathsf{UNF}}_{\mathsf{ABC},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(1^\lambda)$$

Intuition: An ABC system is unforgeable if no probabilistic polynomial-time adversary can produce a valid proof for a predicate that they cannot legitimately satisfy based on credentials they have been issued. This prevents forging credentials or proving false statements about them.

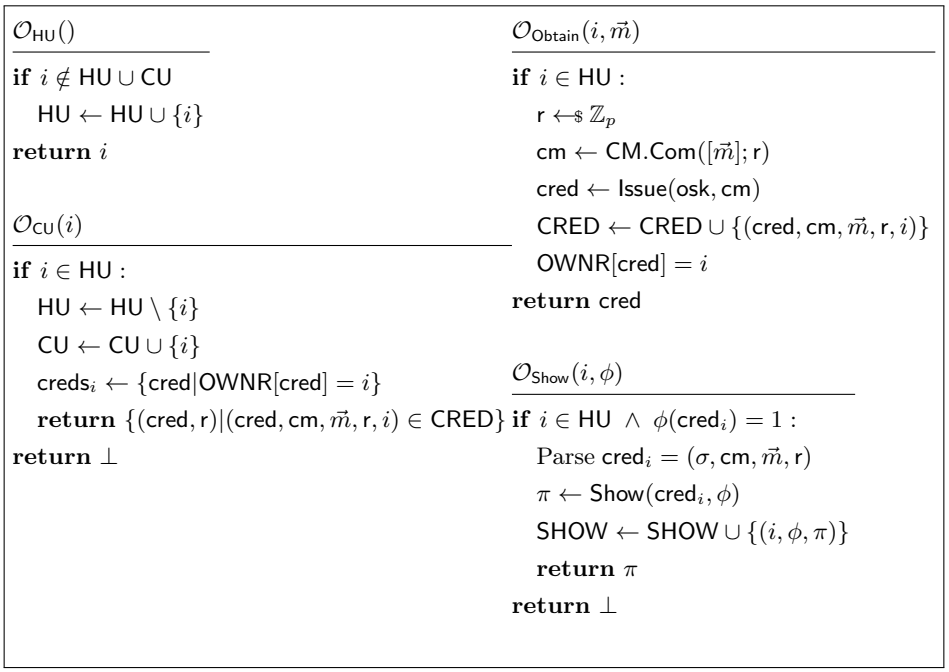

Fig. 2. Oracles for the ABC Security Games

Definition 3.28 (Anonymity). *An* ABC *system provides anonymity if, for all PPT adversaries $\mathcal{A}$, the advantage in the following experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A}}(1^\lambda) = \left|\Pr[\mathrm{Game}^{\mathsf{anon}-1}_{\mathsf{ABC},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathrm{Game}^{\mathsf{anon}-0}_{\mathsf{ABC},\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

Anonymity Intuition: An ABC system provides anonymity if no PPT adversary can determine which user's credential was used in a proof, even if the adversary controls the issuer and chooses the messages and predicates. The challenger sets up the game by picking a random bit $b \leftarrow\!\!\$\ \{0, 1\}$, which determines whether "Alice's or Bob's" credential is used. The adversary's probability of guessing correctly should be negligibly close to random guessing.

3.5.3 Our Construction

Outline Our credential system operates over attribute space $\mathbb{Z}_p$. The user is indexed by i. The credential cred is a rerandomizable Pointcheval-Sanders signature over commitments $\sigma \leftarrow \mathsf{RS.Sign}(\mathsf{cm}, \mathsf{osk})$ where $\mathsf{cm} \leftarrow \mathsf{CM.Com}(\vec{m}; \mathsf{r})$. During verification, the user rerandomizes both signature and commitment for anonymity, then uses Σ-protocols to prove their correctness for any predicate ϕ. This approach leverages the algebraic structure of PS Signatures and Pedersen Commitments, that is, messages are exponents of a commitment which yields well-known, highly expressive and efficient zero-knowledge proofs of group element exponents, supporting a wide range of statements from selective disclosure to complex arithmetic relations. Theoretically, proofs are linear in the size of the attribute vector, however, we prove in 7.3.1 that the practical performance is sub-linear and extremely efficient. In contrast, SPS-EQ [FHS19, CLPK22] use constant-size, efficient set commitment proof with albeit limited expressiveness. On the other hand, [RAR+24] use Groth-Sahai proofs which are expressive but less efficient as each proof is an elliptic curve pairing. During Obtain, Issue, the user sends the commitment cm along with a proof of opening $\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm})$ allowing the extraction of r for corrupt users in the unforgeability proof.

3.5.4 Our Sigma Protocols

Our ABC system relies on five core relations proven via Σ-protocols:

1. **Commitment Opening:** For commitment cm to attribute vector $\mathsf{attrs} = [\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{attrs}]$ and randomness r. $\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm})$ is a proof for relation:

$$\mathcal{R}_{\mathsf{com}} = \mathsf{ZKPoK}\left\{((\mathsf{cm}), (\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}, \mathsf{r}))|\mathsf{cm} = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}}, g_3^{\mathsf{exp}} g_4^{\mathsf{s}}, g^{\mathsf{r}}\right\}$$

2. **Proof of Zero's**: we use adjust the commitment opening proof to prove specific committed values are zero by removing their position in the commitment key. To prove that $\mathsf{cm} = \mathsf{CM.Com}([0, 0, 0, \mathsf{s}]; \mathsf{r})$, we run the commitment opening protocol with $\mathsf{ck} = g_4 g$, if the output verifies, we prove the absence of the first three exponents. $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm})$ is a proof for relation:

$$\mathcal{R}_{\mathsf{zero}} = \{((\mathsf{cm}), (\mathsf{s}_1, \mathsf{r})) \mid \mathsf{cm} = g_1^0 g_2^0 g_3^0 g_4^{\mathsf{s}_1} g^{\mathsf{r}}\}$$

3. **Signature Validity:** after rerandomization, our signatures in the form $\sigma' = (\sigma'_1, \sigma'_2)$ combine pairing verification with sigma protocol to prove knowledge of the committed messages and randomization factor. $\Pi^{\mathcal{R}_\sigma}$ is a proof for relation:

$$\mathcal{R}_\sigma = \mathsf{ZKPoK}\left\{ ((\sigma', \mathsf{cm}'), (\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}, \mathsf{r} + \Delta_{\mathsf{r}})) \middle| \begin{array}{ll} e(\sigma_1, \mathsf{vk} \cdot \widetilde{\mathsf{cm}}) = e(\sigma_2, \tilde{g}) & \wedge \\ e(\mathsf{cm}, \tilde{g}) = e(g, \widetilde{\mathsf{cm}}) & \wedge \\ \mathsf{cm} = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}} g_3^{\mathsf{exp}}, g_4^{\mathsf{s}} g^{\mathsf{r}+\Delta_{\mathsf{r}}} & \end{array} \right\}$$

4. **Malicious Issuer Protection:** For verification key vk and commitment key ck, $\Pi^{\mathcal{R}_{\mathsf{verkey}}}$ is a proof for relation:

$$\mathcal{R}_{\mathsf{verkey}} = \{(\mathsf{vk}, \mathsf{ck}, (\mathsf{sk}, x, \{y_i\}_{i=1}^{\ell})) \mid \mathsf{sk} = g^x \wedge \mathsf{vk} = \tilde{g}^x \wedge \forall i \in [1, \ell] : g_i = g^{y_i} \wedge \tilde{g}_i = \tilde{g}^{y_i}\}$$

This relation proves the issuer knows the discrete logarithms of their public keys, ensuring they can't create malformed keys that might enable deanonymization or signature forgery. The issuer must provide this proof during credential issuance, preventing attacks that exploit maliciously crafted keys with hidden structures.

Example Consider a user holding a passport credential needing to satisfy ϕ such that $\mathsf{ctx} =$ "*master*" $\wedge$ $\mathsf{exp} >$ today.

Passport
id : 12345,
ctx : "*master*",
exp : "10/11/2026"
s : 54321

Fig. 3. Example Master Credential

The user rerandomizes their signature and commitment by generating blinding factors $\Delta_r, \Delta_u \leftarrow\$ \mathbb{Z}_p^2$, computes $\sigma' \leftarrow \mathsf{RS.Rand}(\sigma, \Delta_r, \Delta_u)$ and $\mathsf{cm}' \leftarrow \mathsf{CM.Rand}(\mathsf{cm}, \Delta_r)$ and generates a ZKPoK satisfying

$$\mathcal{R}_\phi = \mathsf{ZKPoK}\left\{ ((\mathsf{cm}', \sigma'), (\mathsf{id}, \mathsf{ctx}, \mathsf{s}, \mathsf{exp}, \mathsf{r} + \Delta_{\mathsf{r}})) \middle| \begin{array}{ll} \mathsf{RS.Ver}(\sigma', \mathsf{cm}', \mathsf{vk}) = 1 & \wedge \\ \mathsf{cm}' = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}}, g_3^{\mathsf{exp}}, g_4^{\mathsf{s}}, g^{\mathsf{r}+\Delta_{\mathsf{r}}} & \wedge \\ \phi(\mathsf{ctx}, \mathsf{exp}) = 1 & \end{array} \right\}$$

Freshness We prevent replay attacks via the challenge phase of our Σ-protocols [CDS94, Dam10]. Verifiers send random challenges that provers must incorporate into their responses. This approach requires no persistent state for verifiers and prevents cross-verifier-proof reuse. Non-interactive versions via Fiat-Shamir [FS86] would require tracking used proofs.

Malicious Organization Keys To prevent attacks from malicious issuers, we extend our signature scheme with:

- $\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) \to \{0,1\}$: Verifies issuer keys via relation:

$$\mathcal{R}_{\mathsf{verkey}} = \{(\mathsf{vk}, \mathsf{ck}), (\mathsf{sk}, x, \{y_i\}_{i=1}^{\ell}) \mid \mathsf{sk} = g^x \wedge \mathsf{vk} = \tilde{g}^x \wedge \bigwedge_{i=1}^{\ell} (g_i = g^{y_i} \wedge \tilde{g}_i = \tilde{g}^{y_i})\}$$

This prevents deanonymization via specially crafted keys, and signature forgery via hidden key relationships. In security proofs, extractability enables reductions to standard cryptographic assumptions.

Two-Party Protocol for VRF Key Generation The key k in the master credential, used as input to a Verifiable Random Function (VRF), is generated through a two-party protocol between the user and issuer to enhance both unforgeability and privacy. The user samples a secret $k_1 \leftarrow\!\$\ \mathbb{Z}_p$ and commits to it, while the issuer samples $k_2 \leftarrow\!\$\ \mathbb{Z}_p$ and embeds it into the credential's attribute vector. The final key, computed as $k = k_1 + k_2$, ensures k can't have been copied from another user or been maliciously issued by the issuer. The issuer learns the user's identity during registration but never learns the complete VRF key, maintaining user privacy while still enabling credential verification.

3.5.5 Security of our Construction

Theorem 3.29 (Unforgeability). *The* ABC *system is unforgeable if the rerandomizable signature scheme is* EUF-CMA *secure, the Pedersen Commitment scheme is position binding, and the* $\Sigma-$*protocol is sound.*

Proof (Sketch). We reduce the security of our ABC system to three underlying properties: EUF-CMA security of the signature scheme, position binding of the commitment scheme, and the soundness of the Σ-protocol. Any successful forgery must break at least one of these properties.

The reduction algorithm $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ receives a PS signature challenge verification key vk and embeds it in the ABC public parameters given to $\mathcal{A}$.For Obtain queries, $\mathcal{B}$ forwards the commitment to the EUF-CMA signing oracle and returns the signature to $\mathcal{A}$. When $\mathcal{A}$ outputs a forgery $(\mathsf{cred}^*, \phi^*, \pi^*)$, $\mathcal{B}$ analyzes it

- **Case 1:** If cred^* contains a signature on a commitment not previously queried, $\mathcal{B}$ outputs this as an EUF-CMA forgery.
- **Case 2:** If cred^* contains a signature on a legitimate commitment but proves a predicate ϕ^* the original attributes don't satisfy:
  - By the special soundness of the Sigma protocol, $\mathcal{B}$ can use a rewinding extractor to extract a valid witness from π^*.
  - This witness must include attribute values that satisfy ϕ^*.
  - Since the original attributes don't satisfy ϕ^*, the extracted witness must differ from the original commitment opening.
  - This gives two different openings for the same commitment, breaking position binding.
- **Case 3:** If the proof π^* verifies but neither Case 1 nor Case 2 applies, this directly contradicts the soundness of the Sigma protocol.

Theorem 3.30 (Anonymity). *The* ABC *system is anonymous even in the case of malicious issuer keys if the rerandomized signature is computationally indistinguishable from the standard, the commitment is hidden, and the zero-knowledge property of the* Σ*-protocol* **??** *ensures a simulator generates* π *indistinguishable from real proofs.*

Fig. 4. ABC System

$\underline{\mathsf{OrgKeyGen}(1^\lambda, 1^\ell)}$ for attribute vector length ℓ

$\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, p) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda),\ \mathsf{ck} \leftarrow\!\$\ \mathsf{CM.Setup}(\mathsf{BG}, 1^\lambda, \ell)$

$(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\$\ \mathsf{RS.KeyGen}(\mathsf{ck}),$ Return $(\mathsf{osk}, \mathsf{opk}) = (\mathsf{sk}, (\mathsf{vk}, \mathsf{ck}))$

$\underline{(\mathsf{Obtain}, \mathsf{Issue})}$:

$\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1) = \mathsf{ZKPoK}\{(\mathsf{s}_1, \mathsf{r}) \mid \mathsf{cm} = g_1^0 g_2^0 g_3^0 g_4^{\mathsf{s}_1} g^{\mathsf{r}}\}$

$\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) = \mathsf{ZKPoK}\{(\mathsf{sk}, x, \{y_i\}_{i=1}^\ell) \mid \mathsf{sk} = g^x \wedge \mathsf{vk} = \tilde{g}^x \bigwedge_{i=1}^\ell (g_i = g^{y_i} \wedge \tilde{g}_i = \tilde{g}^{y_i})\}$

$\underline{\mathsf{Obtain}(\mathsf{opk})}$		$\underline{\mathsf{Issue}(\Pi^{\mathcal{R}_{\mathsf{zero}}}, \mathsf{cm}, \mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{osk})}$
	$\xleftarrow{\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk},\mathsf{vk},\mathsf{ck})}$	Compute and send $\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck})$
If $\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{vk}, \mathsf{ck})$ fails, return $\perp$		
$\mathsf{s}_1 \leftarrow\!\$\ \mathbb{Z}_p, \mathsf{r} \leftarrow\!\$\ \mathbb{Z}_p\ , \mathsf{cm}_1 = \mathsf{CM.Com}([0,0,0,\mathsf{s}_1]; \mathsf{r})$		
	$\xrightarrow{\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)}$	If $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)$ fails, return $\perp$
		$\mathsf{s}_2 \leftarrow\!\$\ \mathbb{Z}_p, \mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}_2]; 0)$ where ctx="master"
		$\mathsf{cm} = \mathsf{cm}_1 + \mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}_1 + \mathsf{s}_2]; \mathsf{r})$
If $\mathsf{RS.Ver}(\sigma, \mathsf{cm}, \mathsf{opk}) = 0$, return $\perp$	$\xleftarrow{\sigma,\mathsf{cm},\mathsf{s}_2,\mathsf{id},\mathsf{ctx},\mathsf{exp}}$	$u \leftarrow\!\$\ \mathbb{Z}_p,\ \sigma \leftarrow\!\$\ \mathsf{RS.Sign}(\mathsf{cm}, \mathsf{osk}, u)$
Else, return $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{r}, \mathsf{opk})$		

$\underline{(\mathsf{Show}, \mathsf{Verify})}$ for credential cred and predicate ϕ:

$\Pi_\phi = \mathsf{ZKPoK}\{(\vec{m}, \mathsf{r}') \mid \mathsf{cm}' = \mathsf{CM.Com}(\vec{m}; \mathsf{r}') \wedge \mathsf{RS.Ver}(\sigma', \mathsf{cm}', \mathsf{opk}) = 1 \wedge \phi(\vec{m}) = 1\}$

$\underline{\mathsf{Show}(\mathsf{cred})}$		$\underline{\mathsf{Verify}(\sigma', \mathsf{cm}', \pi_\phi, \mathsf{opk})}$
		Send empty access policy $\phi = \perp$
Parse $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{r}, \mathsf{opk})$		
Sample $\Delta_\mathsf{r}, \Delta_u \leftarrow\!\$\ \mathbb{Z}_p$		
$\sigma' = \mathsf{RS.Rand}(\sigma, \Delta_\mathsf{r}, \Delta_u)$		
$\mathsf{cm}' = \mathsf{CM.Rand}(\mathsf{cm}, \Delta_\mathsf{r}),\ \mathsf{r}' = \mathsf{r} + \Delta_\mathsf{r}$		
Compute Π_ϕ		
	$\xrightarrow{\sigma',\mathsf{cm}',\pi_\phi}$	If π_ϕ fails, return 0, else 1

Proof (Sketch). We prove anonymity using a hybrid argument that shows the adversary's advantage is negligible. The proof specifically addresses protection against malicious issuers through our key verification mechanism.

Hybrid 0 (Real Game): The challenger follows $\text{Game}^{\mathsf{ANON}}_{\mathsf{ABC},\mathcal{A}}$ exactly. For a randomly chosen $b \in \{0,1\}$, the challenger uses user i_b's credentials $(\mathsf{cred}_{i_b}, \mathsf{cm}_{i_b}, \mathsf{r}_{i_b})$ to generate $(\mathsf{cred}', \mathsf{cm}', \pi)$ via the real Show protocol.

Hybrid 1 (Simulated Proof): The challenger generates cred' and cm' by rerandomizing cred_{i_b} and cm_{i_b} as in Show, but uses a zero-knowledge simulator $\mathcal{S}$ to produce the proof π without using the witness:

$$\begin{aligned} \mathsf{cred}' &\leftarrow \mathsf{RS.Rand}(\mathsf{cred}_{i_b}, \Delta_r, \Delta_u) \\ \mathsf{cm}' &\leftarrow \mathsf{CM.Rand}(\mathsf{cm}_{i_b}, \Delta_r) \\ \pi &\leftarrow \mathcal{S}(\mathsf{cred}', \mathsf{cm}', \mathsf{opk}, \phi) \end{aligned}$$

Indistinguishability of Hybrid 0 and Hybrid 1: By the zero-knowledge property of our Σ-protocol, there exists a simulator $\mathcal{S}$ such that for any $b \in \{0,1\}$, real proofs and simulated proofs are computationally indistinguishable. This holds even with adversarially chosen parameters, provided the issuer proves validity of the verification key using $\Pi^{\mathcal{R}_{\mathsf{verkey}}}$. The difference in the adversary's success probability between Hybrid_0 and Hybrid_1 is negligible.

Independence of b in Hybrid 1: In Hybrid_1, the adversary's view is independent of the bit b. This follows from:

1. **Perfect Hiding of Commitments:** The rerandomized commitment cm' has a uniform distribution regardless of which original commitment was used, due to the perfect hiding property of Pedersen commitments.

2. **Rerandomization of Signatures:** The rerandomized signature cred' has a distribution that is computationally indistinguishable from a fresh signature, due to the rerandomization property of the PS signature scheme.

3. **Simulated Proof Independence:** The simulated proof π depends only on the public statement $(\mathsf{cred}', \mathsf{cm}', \phi)$, not on the witness, making it independent of which user was chosen.

4. **Protection Against Malicious Keys:** The $\Pi^{\mathcal{R}_{\mathsf{verkey}}}$ proof executed during issuance ensures the issuer knows the discrete logarithms of all generators, preventing the embedding of malicious structures that could break unlinkability.

Therefore, $\Pr[\mathcal{A} = 1 | b = 0] = \Pr[\mathcal{A} = 1 | b = 1]$ in Hybrid_1, making the adversary's advantage exactly zero in this hybrid.

The advantage thus bounds the adversary's total advantage in the anonymity game in distinguishing between Hybrid_0 and Hybrid_1, which is negligible under our security assumptions.

3.6 Performance Evaluation

We implemented all credential schemes using the arkworks library [ark22] in Rust, ensuring consistent cryptographic primitives across all implementations. Our experiments were conducted on a MacBook Air M2 (2022) with 16GB RAM running macOS Sequoia 15.3.2. All measurements represent the average of 100 independent trials with standard deviations below 5%.

Implementation Details We used the BLS12-381 elliptic curve group with 381-bit base field (approximately 128-bit security level) for all evaluated schemes. For multi-scalar multiplication operations, we leveraged arkworks' optimized MSM implementation that uses Pippenger's algorithm with windowing optimizations.

Schemes Evaluated We implemented and compared five anonymous credential schemes:

- BBS+ (2006) [ASM06]: The original BBS+ signature scheme
- BBS+ (2016) [CDL16]: An optimized version with improved verification
- PS (2016) [PS16]: The Pointcheval-Sanders signature scheme adapted for credentials
- PS-UTT G1 [TBA+22]: The UTT variant with signatures in $\mathbb{G}_1$
- PS-UTT G2 (Our work): Our optimized variant with signatures in $\mathbb{G}_2$

Performance Metrics For each scheme, we measured four primary operations:

- **Obtain**: User commitment generation and request preparation
- **Issue**: Issuer credential generation and signing
- **Show**: User credential presentation with zero-knowledge proof generation
- **Verify**: Verifier signature and proof verification

We focused on per-credential computation time as our primary metric, varying the number of attributes per credential from 2 to 30 to understand scaling behavior. Each operation was timed independently to identify specific performance bottlenecks and advantages.

Our implementation is available in our open-source repository [Pol25] allowing for reproduction of our results.

3.6.1 Our Construction Improves on Key Credential Operations

Our G2 signature variant **??** represents an optimization over the PS-UTT G1 approach described in [TBA+22]. By relocating signature components from $\mathbb{G}_1$ to $\mathbb{G}_2$, we achieve verification speedups and complete Show + Verify speedup by reducing a pairing operation.

Table 3. Performance Comparison: PS-UTT and Our Construction (time in ms)

Attributes	Show		Verify		Show + Verify	
	G1	G2 (Ours)	G1	G2 (Ours)	G1	G2 (Ours)
2	1.14	1.29 (11.6% ↑)	2.47	1.79 (27.5% ↓)	3.61	3.08 (14.7%↓)
5	1.16	1.29 (10.1% ↑)	2.73	2.01 (26.4%↓)	3.90	3.30 (15.4%↓)
10	1.22	1.33 (8.27% ↑)	3.16	2.44 (22.8%↓)	4.38	3.77 (13.9%↓)
15	1.40	1.37 (2.19% ↓)	3.47	2.72 (21.6%↓)	4.87	4.09 (16.0%↓)
20	1.41	1.51 (6.62% ↑)	3.84	3.21 (16.4%↓)	5.25	4.72 (10.1%↓)
30	1.37	1.59 (13.8% ↑)	4.67	3.79 (18.8%↓)	6.04	5.37 (11.1%↓)

3.6.2 Performance Comparison: Our Construction is State-Of-The-Art

We now present a comprehensive comparison of our construction against four state-of-the-art anonymous credential schemes. Table 7.2.1 summarizes the performance across all credential operations and attribute counts.

Key Findings

1. **Optimized**: Our PS-UTT G2 variant achieves the best Show+Verify performance (10-16% faster than PS-UTT G1 and up to 83% faster than BBS+ 2006), optimizing the operation that occurs

Table 4. Performance of Anonymous Credential Operations (time in ms), n is attribute count

n	[ASM06]	[CDL16]	[PS16]	[TBA$^+$22] ??	Our Construction
			Obtain		
2	0.51	0.90	0.66	0.25	**0.23**
5	0.65	1.00	0.66	0.28	**0.27**
10	0.67	1.13	0.82	0.36	**0.31**
15	0.78	1.26	0.87	0.37	**0.36**
20	0.86	1.38	0.94	**0.41**	**0.41**
30	1.07	1.63	1.11	0.51	**0.49**
			Issue		
2	1.25	**0.72**	1.48	1.27	2.99
5	1.66	**0.75**	1.79	1.66	3.31
10	2.33	**0.83**	2.54	2.35	4.00
15	2.98	**0.84**	3.23	3.03	4.64
20	3.96	**0.90**	3.79	3.66	5.88
30	4.97	**0.94**	5.16	5.10	6.86
			Show		
2	5.39	2.31	3.20	**1.14**	1.29
5	6.05	2.42	3.15	**1.16**	1.29
10	7.44	1.71	4.53	**1.22**	1.33
15	8.86	2.71	6.14	1.40	**1.37**
20	11.88	1.88	7.66	**1.41**	1.51
30	12.91	3.15	16.23	**1.37**	1.59
			Verify		
2	7.59	2.18	4.57	2.47	**1.79**
5	9.25	2.25	5.52	2.73	**2.01**
10	11.09	**2.25**	7.10	3.16	2.44
15	13.96	**2.30**	8.62	3.47	2.72
20	16.93	**2.34**	9.88	3.84	3.21
30	26.30	**2.57**	16.55	4.67	3.79
			Issuing Phase Total (Obtain + Issue)		
2	1.76	1.62	2.14	**1.53**	3.22
5	2.31	**1.76**	2.45	1.95	3.57
10	3.00	**1.96**	3.37	2.71	4.31
15	3.75	**2.10**	4.10	3.40	5.00
20	4.82	**2.29**	4.74	4.06	6.28
30	6.04	**2.57**	6.27	5.60	7.35
			Showing Phase Total (Show + Verify)		
2	12.98	4.48	7.77	3.61	**3.08**
5	15.30	4.67	8.68	3.90	**3.30**
10	18.53	3.96	11.62	4.38	**3.77**
15	22.82	4.22	14.76	4.87	**4.09**
20	28.81	5.01	17.53	5.25	**4.72**
30	39.21	5.72	32.77	6.04	**5.37**

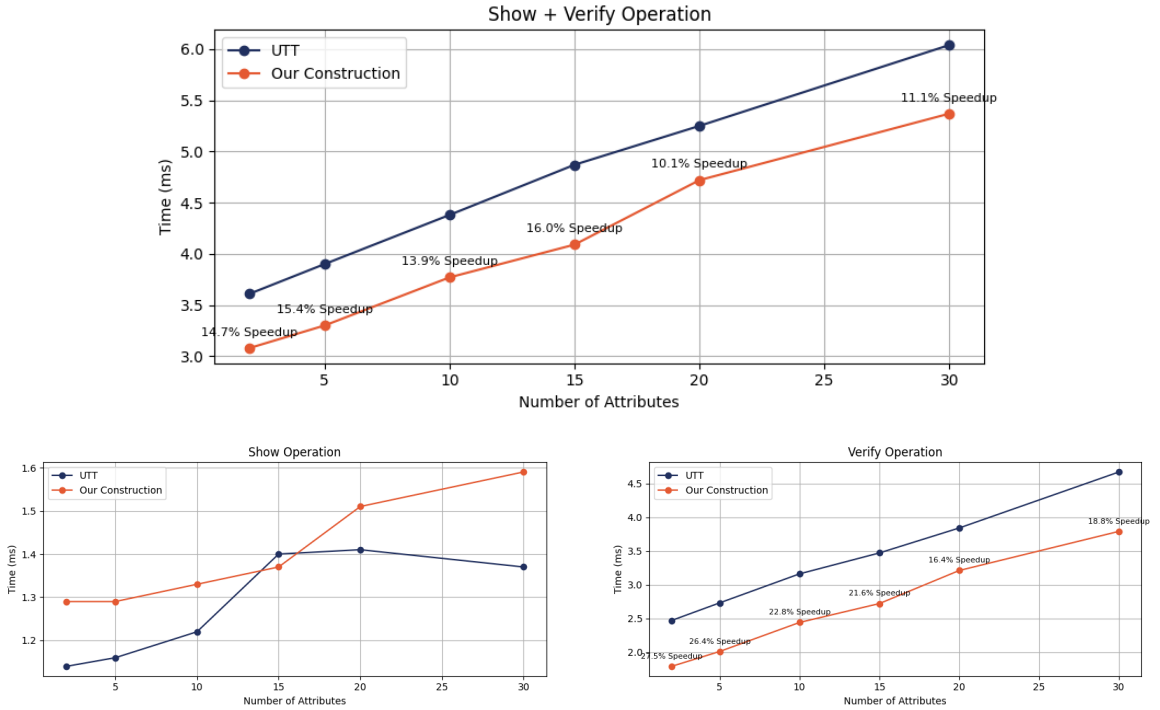

Fig. 5. Performance Comparison between UTT and Our Construction. We improve key operations: Verify and Show + Verify

most frequently in practice. While the Issue operation is more costly, credentials are issued once but verified many times.

2. **Evolution of Efficiency**: The original constructions [ASM06,PS16] required proof of knowledge for $\mathbb{G}_T$ points, resulting in slower operations. Later optimizations [CDL16, TBA+22] shifted proof of knowledge work to $\mathbb{G}_1$ reducing Show+Verify time by up to 6x.

3. **Favorable Scaling Properties**: While all schemes exhibit approximately linear scaling with attribute count, our PS-UTT G2 is the most efficient.

3.6.3 Future Work

Benchmark these schemes against SPS-EQ and zk-creds for efficient anonymous credential comparison.

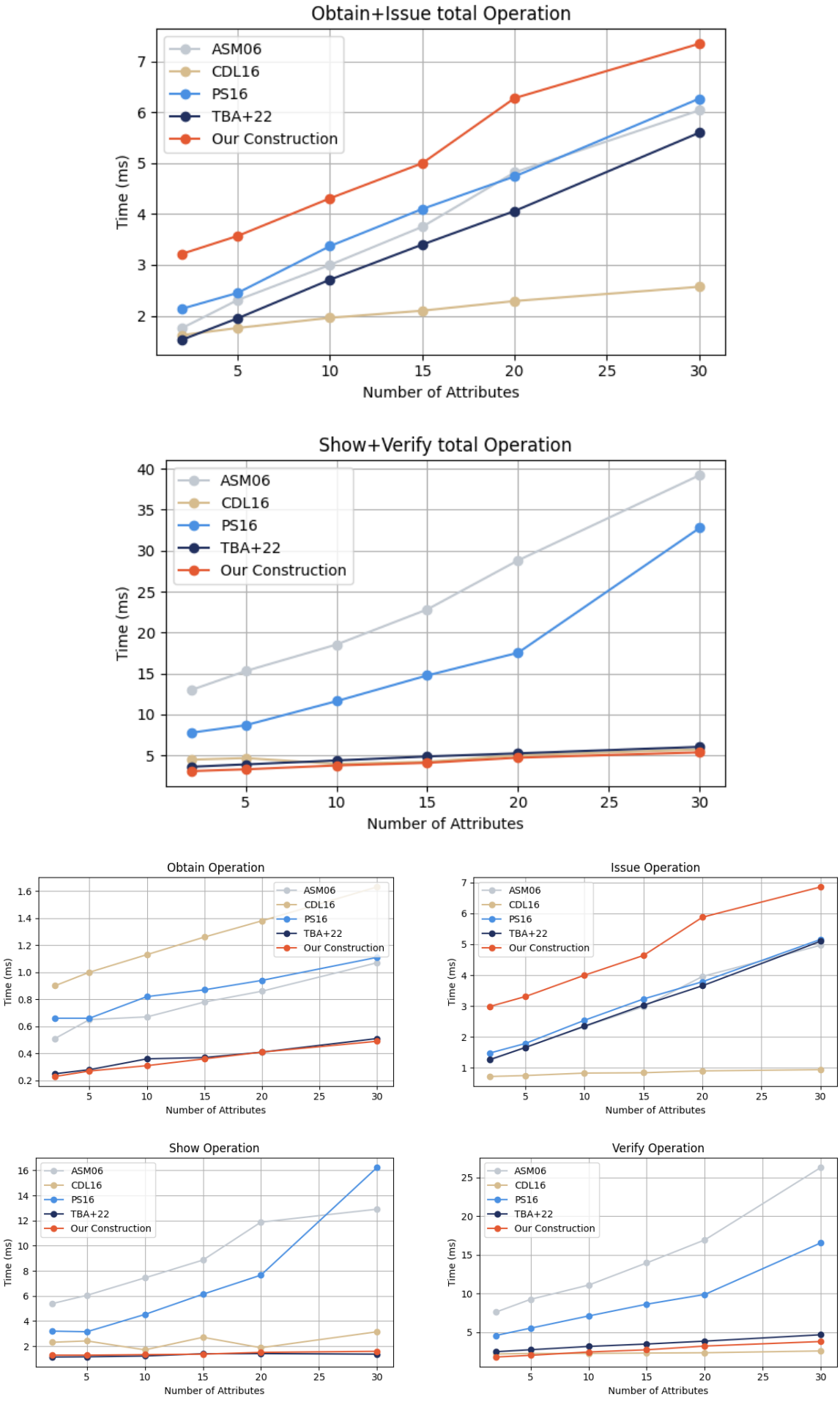

Fig. 6. Performance comparison of different operations across varying numbers of attributes

Chapter 4

Identity Binding Multi Issuer Multi Credential Anonymous Credentials

4.1 Introduction

Anonymous credential systems enable users to prove statements about their attributes while preserving privacy, evolving from single-issuer designs (e.g., Idemix [CVH02]) to meet complex, real-world demands. Unlike Chapter 2's single-issuer ABC system, we address the problem *how can users privately combine credentials from multiple, mutually distrusting issuers to prove they belong to the same identity, without revealing it?*

Consider a user proving they hold (1) a government-issued ID confirming residency, (2) an employer credential verifying income, and (3) a training certificate—all tied to one identity, without linking them via a public identifier. Alternatively, in content credentialing, a user might present images signed by different devices (e.g., cameras) to a journal, proving they share an account while selectively disclosing metadata. Existing approaches fall short: attribute-based signatures lack aggregation [CL03], aggregate signatures assume a single issuer [BLS01, BBS04] or limit revocation flexibility [**?**], and generic zkSNARKs, while expressive, incur high computational costs [RWGM22]. CanDID [**?**] binds credentials via a consistent name, but this offers weaker security against malicious issuers and their system lacks issuer-privacy for internal identifiers. ZKcreds [RWGM22] uses a zkSNARK-based join gadget for multi-credential proofs, yet its proof computation scales poorly, especially for multiple credentials. Our Multi-Issuer Multi-Credential Anonymous Credential system (MIMC-ABC) overcomes these limitations with a secure, efficient solution.

MIMC-ABC employs position-binding commitments and zero-knowledge proofs to cryptographically bind credentials from distinct issuers to a single, private identifier, ensuring anonymity and unforgeability even against colluding adversaries. We formalize a security model for multi-issuer identity binding, define the identity binding property, and provide rigorous proofs of its guarantees. Our comprehensive attack taxonomy—covering forgery, predicate manipulation, and binding attacks—demonstrates robustness, while a scaling analysis shows security holds as issuers and credentials grow. Performance evaluations reveal that privacy-preserving multi-issuer verification, though roughly 3× slower than non-private baselines (e.g., 18.67ms vs. 6.77ms for 4 credentials), remains efficient for practical use.

4.1.1 Contributions

This chapter advances the state of anonymous credentials with:

- A formalized multi-issuer security model with an identity binding property, ensuring credentials from distinct issuers provably belong to the same user without compromising anonymity
- Security proofs showing resilience as the number of issuers and credentials increases.

– A performance evaluation methodology comparing non-private, private single-issuer (with batch verification), and private multi-issuer scenarios, quantifying privacy's overhead.

Sam: must update the identity binding show

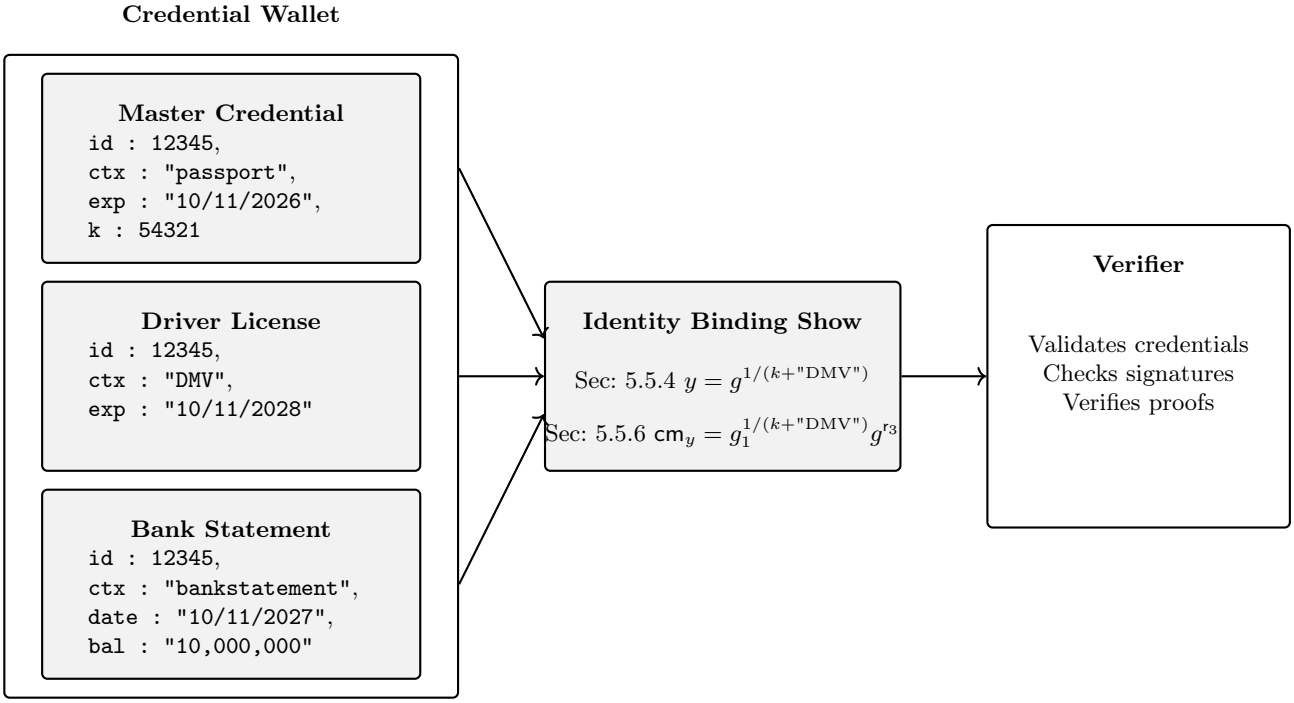

Fig. 7. A credential showing scheme for privacy-preserving protocols, illustrated with a credential hierarchy binding multiple credentials (passport, driver's license, bank statement) to enable secure and privacy-preserving presentation to verifiers.

4.2 System Model and Definitions

Our Multi-Issuer Multi-Credential Anonymous Credential (MIMC-ABC) system extends the single-issuer Attribute-Based Anonymous Credential (ABC) framework from Chapter 2 to support credentials from multiple, mutually distrusting issuers, bound to a single identity. The ABC system (Section 2.4) uses a variant of rerandomizable Pointcheval-Sanders signatures [PS16] and position-binding Pedersen commitments [TBA+22] to enable expressive predicate proofs over attributes. MIMC-ABC builds on this by introducing multi-issuer key generation and a cryptographic identity binding mechanism, ensuring all credentials verifiably belong to the same user without revealing their identity.

4.2.1 MIMC-ABC Syntax

A MIMC-ABC system consists of the following probabilistic polynomial-time (PPT) algorithms, parameterized by a security parameter λ and attribute vector length ℓ:

Definition 4.1 (MIMC-ABC).

– $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: Outputs public parameters* pp*, including a bilinear group* $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, p)$ *as in Section 2.1.*

– $\mathsf{OrgKeyGen}(\mathsf{pp}, \ell, j) \to (\mathsf{osk}_j, \mathsf{opk}_j)$*: For issuer* j*, takes* pp *and* ℓ*, outputs a keypair* $(\mathsf{osk}_j, \mathsf{opk}_j)$ *using the signature scheme from Section 2.3.*

– $(\mathsf{Obtain}(\mathsf{attrs}, \{\mathsf{opk}_j\}_j, \mathsf{aux}), \mathsf{Issue}(\mathsf{osk}_j, \mathsf{cm}, \mathsf{aux})) \to (\mathsf{cred}_j, \perp)$*: An interactive protocol between a user and issuer j. The user inputs an attribute vector* $\mathsf{attrs} = [\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]$*, where* id *is a unique identifier,* ctx_j *is the issuer-specific context, and* attrs_j *are attributes. The user samples* $\mathsf{r}_j \leftarrow\$ \mathbb{Z}_p$*, commits as* $\mathsf{cm} \leftarrow \mathsf{CM.Com}(\mathsf{attrs}; \mathsf{r}_j)$*, and proves its opening (Section 2.4.3). The issuer signs* cm *with* osk_j*, outputting* $\mathsf{cred}_j = (\sigma_j, \mathsf{cm})$ *to the user and* $\perp$ *to itself.*

– $(\mathsf{Show}(\{\mathsf{cred}_j\}_j, \{\mathsf{r}_j\}_j, \phi), \mathsf{Verify}(\{\mathsf{cred}'_j\}_j, \pi)) \to \{0, 1\}$*: An interactive protocol between a user and verifier. The user inputs a set of credentials* $\{\mathsf{cred}_j\}_j$ *from (optionally) distinct issuers, rerandomizes each* (σ_j, cm) *to* $(\sigma'_j, \mathsf{cm}')$*, and computes a proof* π *that* $\phi(\{\mathsf{attrs}_j\}_j) = 1$ *and all* id *values match, using* Σ*-protocols (Section 2.4.3). The verifier checks the proof and rerandomized credentials against* $\{\mathsf{opk}_j\}_j$*, outputting 1 if valid, 0 otherwise.*

4.2.2 Security Properties

MIMC-ABC inherits the core security properties from the ABC system—correctness, unforgeability, and anonymity (Section 2.5)—and adds identity binding for the multi-issuer setting:

– **Correctness**: An honest user with valid credentials from multiple issuers can generate a proof for any predicate ϕ their attributes satisfy, including same-identity constraints, which verifies with probability $1 - \mathsf{negl}(\lambda)$.

– **Unforgeability**: No PPT adversary can produce a valid proof for a predicate ϕ they cannot legitimately satisfy, including forging credentials or mixing credentials from different identities, except with negligible probability.

– **Anonymity**: Proofs reveal only that ϕ is satisfied, not the user's identity or credential details, even if all issuers and the verifier collude.

– **Identity Binding**: When ϕ requires all credentials to share the same id, no PPT adversary can produce a valid proof using credentials with differing id values, except with negligible probability.

4.2.3 Identity Binding Property

We formalize identity binding as a distinct security property for multi-issuer systems:

Definition 4.2 (Identity Binding). *A MIMC-ABC system satisfies identity binding if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\Pr\left[\begin{array}{ll} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) & \mathsf{Verify}(\{\mathsf{cred}'_j\}_j, \phi^*, \pi^*) = 1 \wedge \\ \{(\mathsf{osk}_j, \mathsf{opk}_j)\}_j \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp}, \ell) & : \ \phi^* \text{ requires } \forall j, \mathsf{id}_j = \mathsf{id} \wedge \\ (\{\mathsf{cred}'_j\}_j, \phi^*, \pi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\mathsf{opk}_j\}_j) & \exists j, k : \mathsf{id}_j \neq \mathsf{id}_k \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ *includes oracles for honest user creation, corruption, credential issuance, and proof generation (adapted from Section 2.5).*

This ensures that credentials presented together must share a single, private id, enforced via position-binding commitments and zero-knowledge proofs, as detailed in Section 3.

4.3 Construction

Our MIMC-ABC system constructs credentials as rerandomizable Pointcheval-Sanders signatures from [TBA+22] over position-binding Pedersen commitments, extending the single-issuer ABC system (Section 2.4) to bind credentials from multiple issuers to a shared, private identifier. We leverage the algebraic structure of these primitives and Σ-protocols to prove credential validity and identity consistency efficiently, supporting predicates over an arbitrary number of attributes.

4.3.1 Intuition

Each credential from issuer j commits to an attribute vector $\mathsf{attrs}_j = [\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]$, where id is a unique, user-chosen identifier, ctx_j denotes the credential's context (e.g., "passport"), and exp_j is an expiration date. The vector can extend to additional attributes as needed (e.g., income, degree, date of birth).

Each credential from issuer j commits to an attribute vector $\mathsf{attrs}_j = [\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]$, where id is a unique, user-chosen identifier, ctx_j denotes the credential's context (e.g., "passport"), and exp_j is an expiration date. The vector can extend to additional attributes as needed (e.g., income, degree). Issuance is flexible: the user privately commits to $\mathsf{cm}_1 = \mathsf{CM.Com}([\mathsf{id}, 0, 0]; \mathsf{r}_j)$ with randomness r_j and proves its opening and proves it commits to zero's in positions 2, 3, while issuer j commits to $\mathsf{cm}_2 = \mathsf{CM.Com}([0, \mathsf{ctx}_j, \mathsf{exp}_j]; 0)$ and homomorphically combines them into $\mathsf{cm}_j = \mathsf{cm}_1 \cdot \mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]; \mathsf{r}_j)$, signing it. Alternatively, the user can commit to all attributes privately, and the issuer signs directly, supporting both fully private and issuer-driven scenarios (e.g., credential oracles like DECO [ZMM+20]). To present credentials, the user rerandomizes each signature and commitment, then proves in zero-knowledge that: (1) all signatures are valid under their respective issuer keys, and (2) all commitments share the same id, satisfying a predicate ϕ.

For example, consider a user with credentials from three issuers ($j = 1, 2, 3$): a passport, driver's license, and university degree, each with the same $\mathsf{id} = 12345$ (Figure 8). The user rerandomizes each pair $(\sigma_j, \mathsf{cm}_j)$ to $(\sigma'_j, \mathsf{cm}'_j)$ and proves they meet a policy, e.g., $\phi = (\mathsf{ctx}_1 = \text{"passport"} \wedge \mathsf{exp}_1 > \text{today} \wedge \mathsf{ctx}_2 = \text{"dmv"} \wedge \mathsf{ctx}_3 \in \mathcal{D})$, where $\mathcal{D}$ is a set of accredited universities.

Passport ($j = 1$)	Driver's License ($j = 2$)	University Degree ($j = 3$)
$\mathsf{id} : 12345$,	$\mathsf{id} : 12345$,	$\mathsf{id} : 12345$,
$\mathsf{ctx}_1 :$ "*passport*",	$\mathsf{ctx}_2 :$ "*dmv*",	$\mathsf{ctx}_3 :$ "*usyd − bcompsc*",
$\mathsf{exp}_1 :$ "10/11/2026"	$\mathsf{exp}_2 :$ "05/01/2027"	$\mathsf{exp}_3 :$ "12/31/2024"

Fig. 8. Example credentials from three issuers, sharing $\mathsf{id} = 12345$. Additional attributes (e.g., degree type) can be included.

4.3.2 Construction Details

The MIMC-ABC system operates as follows, reusing primitives from Chapter 2 (Sections 2.3, 2.4):

- $\mathsf{Setup}(1^\lambda)$: Generates pp with bilinear group $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, p)$ and commitment generators $(g_1, g_2, g_3, \tilde{g}_1, \tilde{g}_2, \tilde{g}_3)$ for $\ell = 3$, extensible to more attributes.
- $\mathsf{OrgKeyGen}(\mathsf{pp}, \ell, j)$: Issuer j runs $\mathsf{RS.KeyGen}$ (Section 2.3) to produce $(\mathsf{osk}_j, \mathsf{opk}_j)$, with $\mathsf{opk}_j = (\mathsf{vk}_j, \mathsf{ck}_j)$.
- Obtain and Issue: The user commits $\mathsf{cm}_j = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]; \mathsf{r}_j)$, proves its opening via $\Pi^{\mathcal{R}_{\mathsf{com}}}$ (Section 2.4.3), and sends it to issuer j. Issuer j signs it as $\sigma_j = \mathsf{RS.Sign}(\mathsf{cm}_j, \mathsf{osk}_j)$, returning $\mathsf{cred}_j = (\sigma_j, \mathsf{cm}_j)$. The user retains r_j.
- Show and Verify: For credentials $\{\mathsf{cred}_j\}_j$, the user:
  1. Rerandomizes: $\mathsf{cm}'_j = \mathsf{CM.Rerand}(\mathsf{cm}_j, \Delta_{r_j})$, $\sigma'_j = \mathsf{RS.Rerand}(\sigma_j, \Delta_{r_j}, \Delta_{u_j})$.
  2. Proves via Σ-protocol:

$$\mathcal{R}_\phi = \left\{ (\{\mathsf{cm}'_j, \sigma'_j\}_j, (\mathsf{id}, \{\mathsf{ctx}_j, \mathsf{exp}_j, \mathsf{r}_j + \Delta_{r_j}\}_j)) \;\middle|\; \begin{array}{l} \forall j : \mathsf{RS.Ver}(\sigma'_j, \mathsf{cm}'_j, \mathsf{vk}_j) = 1 \wedge \\ \mathsf{cm}'_j = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}_j} g_3^{\mathsf{exp}_j} g^{\mathsf{r}_j + \Delta_{r_j}} \wedge \\ \phi(\{\mathsf{ctx}_j, \mathsf{exp}_j\}_j) = 1 \end{array} \right\}$$

  The verifier checks π and $\{\mathsf{opk}_j\}_j$, accepting if valid.

4.3.3 Identity Binding Mechanism

Identity binding relies on the position-binding property of Pedersen commitments (Section 2.2). The Σ-protocol proves that all cm'_j share the same id in the first position (g_1^{id}), using an equality proof across commitments:

$$\mathcal{R}_{\mathsf{id}} = \mathsf{ZKPoK}\left\{ (\{\mathsf{cm}'_j\}_j, (\mathsf{id}, \{\mathsf{r}'_j, \mathsf{ctx}_j, \mathsf{exp}_j\}_j)) \,\middle|\, \forall j : \mathsf{cm}'_j = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}_j} g_3^{\mathsf{exp}_j} g^{\mathsf{r}'_j} \right\}$$

The position-binding assumption (SDLP, Section 2.1) ensures an adversary cannot forge commitments with different id values that appear equal, reducing security to standard cryptographic hardness.

4.4 Security Analysis

We analyze the security of MIMC-ABC against a PPT adversary controlling issuers, corrupting users, and querying oracles (adapted from Section 2.5). Our system inherits the ABC framework's guarantees (Section 2.6) and strengthens them with identity binding for the multi-issuer setting. We prove correctness informally, then formalize unforgeability, anonymity, and identity binding, reducing security to the underlying primitives' assumptions: EUF-CMA of the signature scheme (Section 2.3), position-binding of the commitment scheme (Section 2.2), and soundness of Σ-protocols (Section 2.4.3).

4.4.1 Correctness

An honest user with valid credentials $\{\mathsf{cred}_j\}_j$ from issuers $\{j\}$ can always generate a proof π for a predicate ϕ their attributes $\{\mathsf{attrs}_j\}_j = \{[\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]\}_j$ satisfy, including same-id constraints. Rerandomization ensures signatures and commitments verify under $\{\mathsf{opk}_j\}_j$, and the Σ-protocol proves ϕ and id equality with probability $1 - \mathsf{negl}(\lambda)$, following Section 2.6's correctness argument extended to multiple issuers.

4.4.2 Unforgeability

Theorem 4.3 (Unforgeability). *MIMC-ABC is unforgeable if the rerandomizable signature scheme is EUF-CMA secure, the Pedersen commitment scheme is position-binding, and the Σ-protocol is sound. For any PPT adversary $\mathcal{A}$, $\mathcal{A}^{\mathsf{UNF}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.*

Proof (Sketch). We reduce unforgeability to three cases, adapting Section 2.6's ABC proof:

1. **Forged Signature**: If $\mathcal{A}$ outputs a valid $\mathsf{cred}'_j = (\sigma'_j, \mathsf{cm}'_j)$ not issued by any osk_j, we extract σ'_j as an EUF-CMA forgery.

2. **Predicate Misuse**: If $\mathcal{A}$ uses valid credentials but proves a false ϕ^* (e.g., $\mathsf{exp}_j <$ today when $\mathsf{exp}_j >$ today), the Σ-protocol's special soundness lets us extract a witness contradicting the original attributes, breaking position-binding.

3. **Identity Mixing**: If $\mathcal{A}$ combines credentials with different id values yet proves same-id, we extract two openings of some cm'_j (e.g., $g_1^{\mathsf{id}_1}$ vs. $g_1^{\mathsf{id}_2}$), breaking position-binding.

A reduction $\mathcal{B}$ simulates the game (Section 2.5), embedding EUF-CMA and position-binding challenges into $\{\mathsf{opk}_j\}_j$ and cm_j. Any forgery violates one assumption, bounding $\mathcal{A}$'s advantage by $\mathsf{negl}(\lambda)$.

4.4.3 Anonymity

Theorem 4.4 (Anonymity). *MIMC-ABC is anonymous, even against colluding issuers, if the signature scheme's rerandomization is indistinguishable, the commitment scheme is hiding, and the Σ-protocol is zero-knowledge. For any PPT adversary $\mathcal{A}$, $\mathcal{A}^{\mathsf{ANON}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.*

Proof (Sketch). We use a hybrid argument, extending Section 2.6's ABC anonymity proof:

– **Hybrid 0**: Real game with user i_b's credentials $\{\mathsf{cred}_j\}_j$, rerandomized and proven via Show.

– **Hybrid 1**: Replace π with a simulated proof using the Σ-protocol's zero-knowledge simulator.

The hiding property of Pedersen commitments ensures cm'_j is uniform, signature rerandomization makes σ'_j indistinguishable from fresh signatures, and the simulated π hides i_b, even if all issuers share $\{\mathsf{osk}_j\}_j$. The advantage is negligible as hybrids are computationally indistinguishable.

4.4.4 Identity Binding

Theorem 4.5 (Identity Binding). *MIMC-ABC satisfies identity binding under the SDLP assumption (Section 2.1). For any PPT adversary $\mathcal{A}$ mixing credentials with distinct id values, $\mathcal{A}^{\mathsf{BIND}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda) \leq n \cdot m \cdot \mathcal{A}^{\mathsf{SDLP}}(\lambda) + \mathsf{negl}(\lambda)$, where n is the number of issuers and m is credentials per user.*

Proof (Sketch). If $\mathcal{A}$ outputs $\{\mathsf{cred}'_j\}_j$, ϕ^* requiring same id, and π^* verifying despite $\mathsf{id}_j \neq \mathsf{id}_k$ for some j, k, the Σ-protocol's special soundness extracts witnesses from π^*. For each $\mathsf{cm}'_j = g_1^{\mathsf{id}_j} g_2^{\mathsf{ctx}_j} g_3^{\mathsf{exp}_j} g^{\mathsf{r}'_j}$, we get id_j, and differing id values imply two openings of some cm'_j at position 1 (e.g., $g_1^{\mathsf{id}_1}$ vs. $g_1^{\mathsf{id}_2}$). A reduction $\mathcal{B}$ embeds an SDLP challenge $(g^x, \tilde{g}^x)$ into $g_1, \tilde{g}_1$, solving x with probability $1/(n \cdot m)$ per credential, yielding the bound.

4.4.5 Security Scaling

The identity binding advantage scales linearly with n and m, reflecting the number of opportunities to break position-binding. This graceful degradation ensures MIMC-ABC remains secure as the system grows, a key advantage over prior multi-issuer schemes lacking formal binding guarantees.

4.5 Identity Binding Application: Efficient KYC/AML Identity Verification

Sam: Sam to do

Speak about the KYC/AML process and why identity binding is necessary

4.6 Performance Evaluation

We evaluate MIMC-ABC to quantify the overhead of privacy-preserving multi-issuer credential verification, comparing it against non-private and single-issuer baselines. Unlike prior multi-issuer systems (e.g., CanDID [MMZ+20], ZKcreds [RWGM22]), MIMC-ABC balances strong identity binding with practical efficiency, leveraging our signature optimization **??**. Our benchmarks focus on verification time—the critical path in authentication—across three scenarios: non-private, private single-issuer with batch signature aggregation, and private multi-issuer.

4.6.1 Methodology

We implemented MIMC-ABC using the arkworks library [ark22] on a BLS12-381 curve (128-bit security), running on a MacBook Air M2 (16GB RAM, macOS Sequoia 15.3.2). We measure verification time for 4, 16, and 32 credentials, each with 4 attributes, averaging 100 trials (standard deviation $<$ 5%). Scenarios include:

1. **Non-Private**: Signatures verified with batch aggregation, revealing attrs_j in the clear—a common baseline for non-anonymous systems.

2. **Private Single-Issuer**: All credentials from one issuer, using batch verification of PS signatures (Section 2.3) and a Σ-protocol for same-id and predicate satisfaction.

3. **Private Multi-Issuer**: Each credential from a different issuer, requiring individual signature verification and a Σ-protocol for identity binding (worst-case scenario).

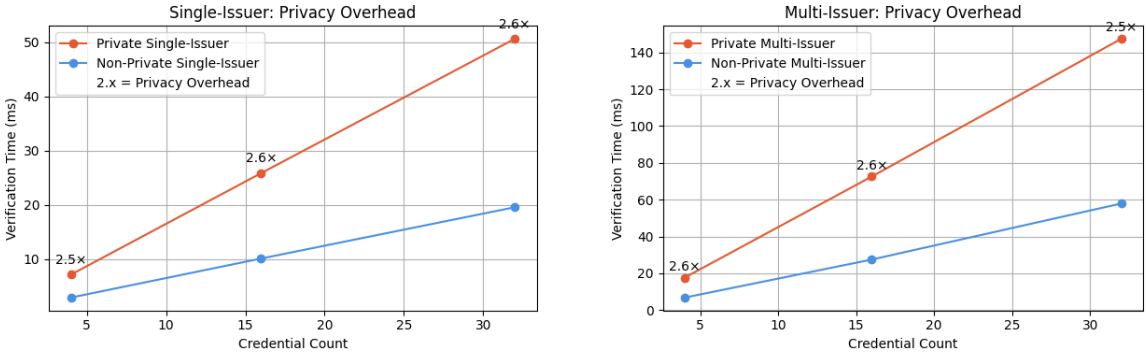

Fig. 9. Performance Comparison between UTT and Our Construction. We improve key operations: Verify and Show + Verify

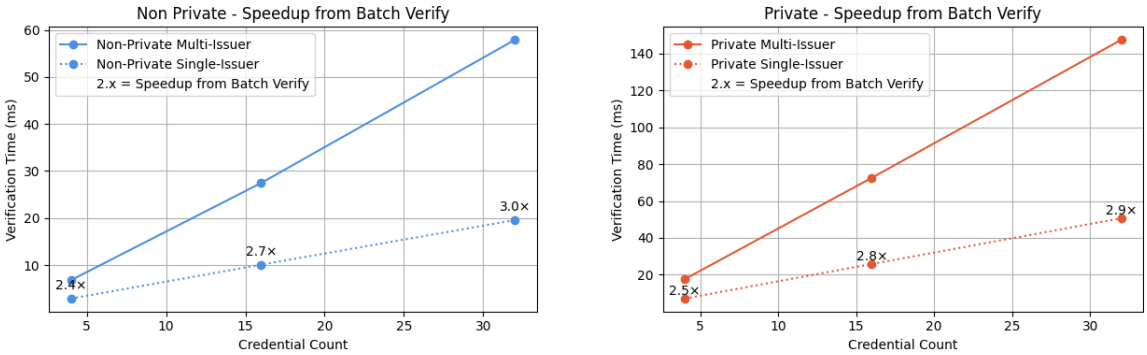

Fig. 10. Performance Comparison between UTT and Our Construction. We improve key operations: Verify and Show + Verify

Table 5. Varying Credential Count, Set Attribute Number (4) (time in ms)

Scenario	Verify (ms)			Overhead Avg.	Notes
	4 Creds	16 Creds	32 Creds		
Non-Private Single-Issuer	2.87	10.08	19.55	–	Cleartext, batched
Private Single-Issuer	7.11	25.85	50.64	2.5×	Batch verification
Non-Private Multi-Issuer	6.79	27.45	57.88	–	Cleartext, distinct issuers
Private Multi-Issuer	17.65	72.57	147.35	2.6×	Worst-case, distinct issuers

Table 6. Varying Attribute Count, Set Credential Count (4) (time in ms)

Scenario	Verify (ms)			Overhead	Notes
	4 Attrs	16 Attrs	32 Attrs	(32 vs. 4 Attrs)	
Non-Private Single-Issuer	2.87	2.85	2.87	1.0×	Cleartext, batched
Private Single-Issuer	7.11	7.28	8.03	1.1×	Batch verification
Non-Private Multi-Issuer	6.79	6.77	6.81	1.0×	Cleartext, distinct issuers
Private Multi-Issuer	17.65	18.67	19.89	1.1×	Worst-case, distinct issuers

4.6.2 Discussion

Table 5 highlights the impact of credential scaling. Non-private single-issuer verification benefits from batch aggregation, scaling near-linearly (2.87ms to 19.55ms). Private single-issuer verification adds a consistent 2.5× overhead across credential counts, remaining efficient due to batching. Non-private multi-issuer verification scales from 6.79ms to 57.88ms, reflecting the cost of individual signature checks. Private multi-issuer verification, at 17.65ms to 147.35ms, incurs a 2.6× overhead at 4 credentials, increasing slightly to 2.5× at 32 credentials, as the lack of batching dominates.

Table 6 examines attribute scaling for 4 credentials. Increasing attributes from 4 to 32 has minimal impact on non-private scenarios (e.g., 2.87ms to 2.87ms for single-issuer, 6.79ms to 6.81ms for multi-issuer), as attribute processing is lightweight without privacy. For private scenarios, the overhead is modest: private single-issuer grows from 7.11ms to 8.03ms (1.1×), and private multi-issuer from 17.65ms to 19.89ms (1.1×). This small increase reflects the efficiency of Σ-protocols in handling additional attributes, a key advantage of MIMC-ABC over zkSNARK-based systems like ZKcreds, which scale poorly with attribute complexity.

In practice, users often hold credentials from a mix of issuers—some repeated, some unique. For example, a user might present two government credentials (batchable) and two from distinct employers, blending single- and multi-issuer efficiency. The worst-case multi-issuer scenario is thus unlikely in full, making MIMC-ABC's average-case performance even more competitive.

Chapter 5

New Nullifier Constructions from the q-DDHI and Applications to Accountable Privacy Systems

Anonymous credential systems enable users to prove identity attributes while preserving privacy. In Chapters 2 and 3, we progressed from single-issuer Attribute-Based Anonymous Credentials (ABC) to a Multi-Issuer Multi-Credential (MIMC) system that securely binds credentials from multiple issuers to a single identity. However, practical privacy-preserving protocols, including credential systems, require mechanisms to enforce uniqueness (e.g., preventing sybil attacks) and revocability without compromising anonymity. Nullifiers—cryptographic primitives that represent a user's action or identity in secrecy—are critical for these purposes, with applications spanning credential binding, anonymous voting, and pseudonymous messaging. This chapter addresses the challenge of designing efficient, privacy-preserving nullifiers using pairing-free Verifiable Random Functions (VRFs), focusing on their application to hierarchical credential binding in the MIMC system while demonstrating their broader utility. Existing VRF constructions, reliant on pairings or RSA, suffer from computational overhead or limited privacy, making them impractical for large-scale anonymous systems. We overcome these limitations with novel Sigma-protocols and prime-order VRFs, achieving significant efficiency and privacy gains.

Chapter Roadmap

The remainder of the chapter is structured as follows: In Section 5.1 we introduce nullifiers and their role in privacy-preserving protocols. In Section 5.2, we introduce the preliminaries and building blocks, in Section 5.5.4, we present our Prime Order DY VRF construction, followed by privacy-preserving extensions in Section 5.5. In Section 5.6, we evaluate our performance against the state-of-the-art schemes, and in Section 5.7, we outline instantiations of our construction for Anonymous Credentials.

5.1 Introduction

Nullifiers are a cornerstone of privacy-preserving protocols, enabling users to perform actions exactly once (e.g. spending a coin, voting, issuing a credential) while remaining anonymous. In real-world identity systems, credentials often form hierarchies—for example, a passport (master credential) with a secret key k serves as a foundational identity, while a driver's license (context credential) is tied to a specific domain ctx (e.g., "DMV"). Binding these credentials to prevent sybil attacks (e.g., obtaining multiple driver's licenses) or enable revocation requires nullifiers that are unique, verifiable, and privacy-preserving. In our Multi-Issuer Multi-Credential (MIMC) system, such nullifiers ensure secure credential management across issuers. Beyond credentials, nullifiers are critical for applications like persistent pseudonymous identities, privacy e-cash systems, anonymous voting, as seen in recent privacy-preserving frameworks [GG22, GNP$^+$15, GKB20, TBA$^+$22, BSCG$^+$14]. Our nullifier scheme takes a master credential key `s` and context `ctx` to produce deterministic outputs for sybil resistance

(e.g., $y = g^{1/(k+ctx)}$) and probabilistic outputs for revocation (e.g., $\mathsf{cm}_y = g_1^{1/(k+ctx)} g^r$), verified via zero-knowledge proofs. Unlike pairing-based schemes [TBA+22], our approach leverages prime-order groups and Sigma-protocols for efficiency and compatibility with standard cryptographic assumptions, making it a versatile and efficient for privacy frameworks.

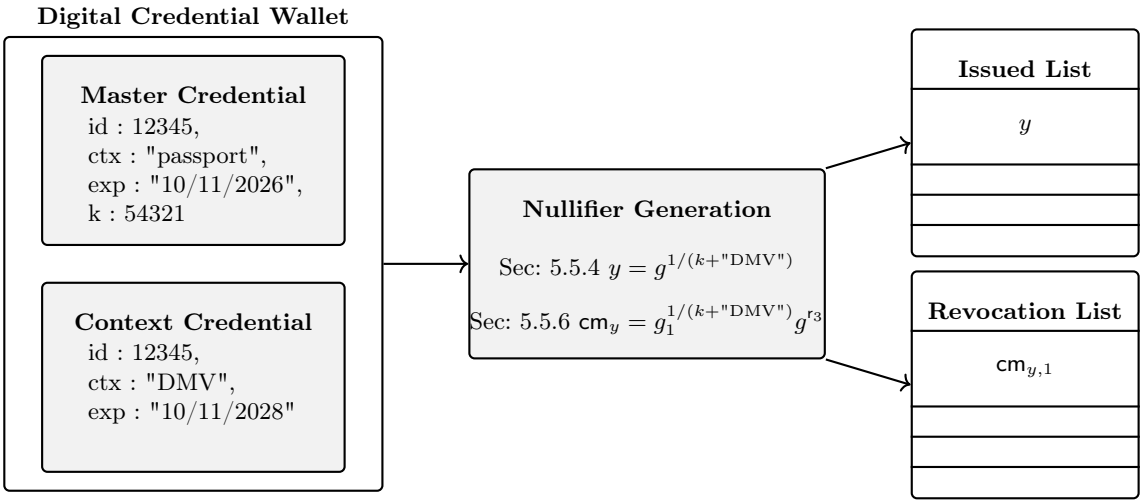

Fig. 11. A nullifier scheme for privacy-preserving protocols, illustrated with a credential hierarchy binding context credentials (e.g., driver's license) to a master credential (e.g., passport). The deterministic nullifier prevents sybil attacks, while the probabilistic nullifier supports revocation, with applications extending to voting, e-cash, and pseudonymous identities

.

5.1.1 Goals

1. **Efficiency**: The nullifier must be generated and verifiable with minimal overhead, avoiding bilinear pairings and MPC while maintaining provable uniqueness and verifiable pseudorandomness.

2. **Anonymity**: Nullifier generation and verification must not reveal the user inputs s, ctx and optionally enable unlinkable outputs.

3. **Integration with Anonymous Credentials**: The nullifier mechanism must seamlessly extend our existing anonymous credential framework without compromising its security properties.

5.1.2 Core Challenges

Creating an efficient, privacy-preserving nullifier for credential binding raises three specific challenges:

1. **Prime Order (Pairing-Free) DY VRF**: The DY VRF provides a secure structure but uses expensive bilinear pairings. Our first challenge is to create a Pairing-Free DY VRF using a Σ-protocol to maintain security under the q-DDHI assumption.

2. **Anonymous Input for Prime-Order VRF**: To transform our Prime-Order DY VRF for Anonymous Credentials, we must use a commitment to the VRF inputs. Our second core challenge is to create a Σ-protocol to prove knowledge of x such that $y = g^{1/x}$ (the q-DDHI challenge). Our third core challenge extends this by incorporating the proof of linear relations, the final protocol proves knowledge of (x, sk) such that $y = g^{1/(sk+x)}$ where y is a deterministic nullifier.

3. **Unlinkable Output for Prime-Order VRF**: The deterministic nullifier can be used for sybil resistance during credential generation but can't be used ongoing as each output links the user's

actions. Our fourth and last core challenge is to extend our protocol to generate provable, probabilistic nullifiers. We need to output a commitment to the nullifier so it can be used without linking the user. The final protocol proves knowledge of (x, sk, r) such that $\mathsf{cm}_y = g_1^{1/(sk+x)} g^r$.

5.1.3 Related Work

We use standard Σ-protocols for their superior efficiency, familiarity among practical implementations, and composability with various cryptographic primitives like our anonymous credential system; additionally, they avoid the computational overhead of MPC and ZK-SNARKs.

Table 7. Comparison of Nullifier/VRF Schemes for Credential Binding

Scheme	Deterministic Output[1]	Unlinkable Output[2]	Private Input[3]	Pairing-Free	Proof Type	Relative Ver. Time[4]
NSEC5 [GNP+15]	✓	✗	✗	✓	Sigma Only	3x faster
PLUME [GG22]	✓	✗	✓	-	ZK-SNARK	Slower
DY VRF [DY05]	✓	✗	✗	✗	Pairing	3x slower
CanDID [MMZ+20]	✓	✗	✓	-	MPC	Very slow [5]
TACT/S3ID [RAR+24]	✓	✗	✓	✗	Groth Sahai + Pairing	4x slower
SyRA [CKS24]	✓	✗	✗	✗	Sigma+Pairing	4x slower
UTT Nullifier [TBA+22]	✓	✓	✓	✗	Sigma+Pairing	4x slower
Sec: 5.5.4:	✓	✗	✓	✓	Sigma only	2x faster
Sec: 5.5.6:	✓	✓	✓	✓	Sigma only	1x (baseline)

[1]Ensures a unique underlying nullifier value per user-context pair for sybil resistance.
[2]Nullifier can be presented as a commitment for unlinkability.
[3]Operates on inputs (e.g., secret keys, attributes) hidden in commitments.
[4]Approximate verification time relative to CRBN, based on benchmarks in Section 5.6
[5]CanDIDs nullifier uses MPF-PRF with uncomparable efficiency [6]

Previous systems have addressed aspects of hierarchical credential binding and sybil resistance, but all have significant limitations:

- **Standard VRFs** [DY05, GNP+15] could generate unique nullifiers but use pairings and reveal the user's public key during verification, compromising anonymity.
- **Pairing-based systems** like SyRA [CKS24] and S3ID [RAR+24] implement hierarchical credentials but suffer from efficiency issues due to their reliance on pairings and S3ID uses Groth Sahai proofs which are less efficient than Σ-protocols.
- **UTT** [TBA+22] uses a similar approach to ours for anonymous payments, creating serial numbers (nullifiers) from a registration credential. However, UTT relies on bilinear pairings, introducing substantial computational overhead.
- **CanDID** [MMZ+20] clearly defines the master/context credential relationship but compromises privacy by maintaining mappings between credential public keys, enabling linkability across interactions.

5.1.4 Contributions

We start with the q-DDHI assumption and work in two different directions. Firstly, we aim for efficiency and reconstruct the Dodis Yampolskiy VRF in a prime-order group and show XXX efficiency improvement while retaining security properties. Secondly, we aim for privacy, starting with a novel zkpok protocol to prove the DDHI challenge ($g^{1/x}$) which we generalise to a sigma protocol for *proof of inverse exponent*. We then extend this with a new private VRF to generate deterministic nullifiers from private inputs, combining our prime-order DY VRF and the ZKPoK of a DDHI challenge,

where inputs are exponents in commitments for use in sybil-resistant mechanisms. We then extend this further for probabilistic outputs, committed but provable nullifiers to be used in revocation schemes.

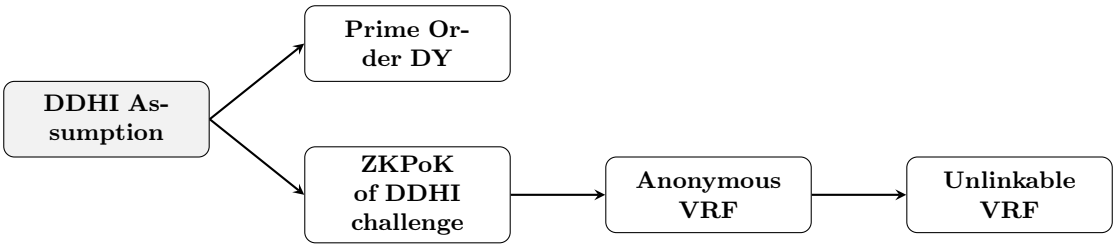

Fig. 12. Sigma Protocols and VRF Constructions from the q-DDHI assumption

5.1.5 Our Novel Sigma Protocols

Table 8. Our Sigma Protocols and Their Applications

Sigma Protocol	Application
DY Prime Order Proof Protocol 5.4.2	Efficient VRF
Proving Knowledge of Committed Inverse Exponent 5.5.2	Generalized Σ-protocol
Proving Knowledge of Committed Inverse Linear Relation 5.5.3	Anonymous Deterministic Nullifier VRF 5.5.4
Proving Knowledge of Committed Nullifier 5.5.5	Anonymous Probabilistic Nullifier VRF 5.5.6

5.2 Preliminaries

5.2.1 Cryptographic Assumptions

Definition 5.1 (q-DDHI Assumption). *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. The q-Decisional-Diffie-Hellman Inversion (q-DDHI) [MSK02] assumption states that for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$\left|\Pr\left[x \leftarrow\$ \mathbb{Z}_p^*, b \leftarrow\$ \{0,1\}, z_0 = g^{1/x}, z_1 \leftarrow\$ \mathbb{G} : \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, z_b) = b\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Informally, given $(g, g^x, g^{x^2}, \ldots, g^{x^q})$, no PPT adversary can distinguish $g^{1/x}$ from a random group element with non-negligible advantage.

Definition 5.2 (q-DBDHI Assumption). *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of prime order p with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and generators $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$. The q-Decisional-Bilinear Diffie-Hellman Inversion (q-DBDHI) assumption states that for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$\left|\Pr\left[x \leftarrow\$ \mathbb{Z}_p^*, b \leftarrow\$ \{0,1\}, z_0 = e(g,\tilde{g})^{1/x}, z_1 \leftarrow\$ \mathbb{G}_T : \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}, \tilde{g}, z_b) = b\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Informally, no PPT adversary can distinguish $e(g,\tilde{g})^{1/x}$ from a random element in $\mathbb{G}_T$ given $(g, g^x, \ldots, g^{x^q}, \tilde{g})$ with non-negligible advantage.

5.2.2 Verifiable Random Functions

A Verifiable Random Function (VRF) [MRV99, DY05] is a pseudorandom function that provides proofs of correct evaluation. Following [Bit20], a VRF consists of these algorithms:

Definition 5.3 (Verifiable Random Function). *A VRF is a tuple of PPT algorithms* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Prove}, \mathsf{VRF.Verify})$ *with message space* $\mathcal{X}$*, output space* $\mathcal{Y}$ *and proof space* Π *where:*

- $\mathsf{VRF.Gen}(1^\lambda) \to (sk, pk)$*: Generates a secret key* sk *and public key* pk*.*
- $\mathsf{VRF.Eval}(sk, x) \to y$*: Computes the VRF output* $y \in \mathcal{Y}$ *for input* $x \in \mathcal{X}$ *using secret key* sk*.*
- $\mathsf{VRF.Prove}(sk, x) \to \pi$*: Produces a proof* $\pi \in \Pi$ *that* $y = \mathsf{VRF.Eval}(sk, x)$ *is computed correctly.*
- $\mathsf{VRF.Verify}(pk, x, y, \pi) \to \{0, 1\}$*: Verifies that* y *is the correct VRF output for input* x *using proof* π*.*

A secure VRF must satisfy the following properties:

- **Completeness:** Honest evaluation and proof generation always passes verification:

$$\Pr\left[\mathsf{VRF.Verify}(pk, x, y, \pi) = 1 \;\middle|\; \begin{array}{l}(sk, pk) \leftarrow \mathsf{VRF.Gen}(1^\lambda) \\ y = \mathsf{VRF.Eval}(sk, x) \\ \pi \leftarrow \mathsf{VRF.Prove}(sk, x)\end{array}\right] = 1$$

- **Uniqueness:** For each input x and public key pk, only one output y can be verified:

$$\text{if} \quad \mathsf{VRF.Verify}(pk, x, y_0, \pi_0) = \mathsf{VRF.Verify}(pk, x, y_1, \pi_1) = 1 \quad \text{then} \quad y_0 = y_1$$

- **Pseudorandomness:** The VRF output is indistinguishable from random for any input not previously queried, defined by the following game $\mathcal{G}^{\text{vrf}}_{\mathcal{A}}$:

  1. The VRF challenger samples $(sk, pk) \leftarrow \mathsf{VRF.Gen}(1^\lambda)$, and sends pk to $\mathcal{A}$.
  2. $\mathcal{A}$ submits evaluation queries $x_1, \ldots, x_Q \in \mathcal{X}$, and receives (y_i, π_i) for each query, where $y_i = \mathsf{VRF.Eval}(sk, x_i)$ and $\pi_i \leftarrow \mathsf{VRF.Prove}(sk, x_i)$.
  3. At any point, $\mathcal{A}$ submits a challenge input $x_* \in \mathcal{X}$ such that $x_* \notin \{x_1, \ldots, x_Q\}$.
  4. The challenger computes $y_0^* = \mathsf{VRF.Eval}(sk, x_*)$, samples $y_1^* \leftarrow\!\!\$\ \mathcal{Y}$ uniformly at random, then samples $b \leftarrow\!\!\$\ \{0, 1\}$ and sends y_b^* to $\mathcal{A}$.
  5. $\mathcal{A}$ may continue to make evaluation queries for inputs other than x_*.
  6. At the end, $\mathcal{A}$ outputs a guess b'. The game outputs 1 if $b' = b$, and 0 otherwise.

  We say that the VRF satisfies pseudorandomness if for all PPT adversaries $\mathcal{A}$:

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

In our work, we focus on adapting the Dodis-Yampolskiy VRF [DY05], which computes $y = e(g, \tilde{g})^{1/(sk+x)}$ in bilinear groups, to work efficiently in standard prime-order groups without pairings.

Definition 5.4 (Dodis Yampolskiy VRF). *The Dodis-Yampolskiy (DY) VRF [DY05] operates in a bilinear group setting with groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *of prime order* p*, and a Type-3 pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*. Let* $g \in \mathbb{G}_1$ *and* $\tilde{g} \in \mathbb{G}_2$ *be generators. The VRF is defined as:*

- $\mathsf{VRF.Gen}(1^\lambda)$: *Sample* $sk \leftarrow\!\$\, \mathbb{Z}_p^*$, *set* $pk = g^{sk}$.
- $\mathsf{VRF.Eval}(sk, x)$: *Compute* $y = e(g, \tilde{g})^{1/(sk+x)}$.
- $\mathsf{VRF.Prove}(sk, x)$: *Compute* $\pi = \tilde{g}^{1/(sk+x)}$.
- $\mathsf{VRF.Vfy}(pk, x, y, \pi)$: *Check* $e(g^x \cdot pk, \pi) \stackrel{?}{=} e(g, \tilde{g})$ *and* $y \stackrel{?}{=} e(g, \pi)$.

Security relies on the q-DBDHI assumption, ensuring y is pseudorandom.

5.2.3 Sigma Protocols and Zero Knowledge Proofs

Our credential binding mechanism relies on zero-knowledge proofs, particularly Sigma-protocols, to verify relations between committed values without revealing them.

A Sigma-protocol is a three-move interactive proof system where:

1. The prover $\mathcal{P}$ sends a commitment message a.
2. The verifier $\mathcal{V}$ sends a random challenge e.
3. The prover responds with z, and $\mathcal{V}$ accepts if the verification equation holds.

These protocols satisfy:

- **Completeness**: For all $(x, w) \in \mathcal{R}$, an honest prover always convinces the verifier.
- **Special Soundness**: There exists an efficient extractor $\mathcal{E}$ such that, given any statement x and two accepting transcripts (a, e, z) and (a, e', z') with $e \neq e'$, $\mathcal{E}$ can extract a witness w such that $(x, w) \in \mathcal{R}$.
- **Special Honest-Verifier Zero-Knowledge**: There exists an efficient simulator $\mathcal{S}$ that, given a statement x and a challenge e, produces a transcript (a, e, z) that is computationally indistinguishable from a real transcript between an honest prover and verifier, without using a witness.

5.2.4 Pedersen Commitment Scheme

We use position-binding Pedersen Commitments from Chapter 2, which allow committing to a vector of messages while hiding the values. For a message vector $[\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}]$ and randomness r, the commitment is:

$$\mathsf{cm} = \mathsf{CM.Com}([m_1, \ldots, m_n]; \mathsf{r}) = g_1^{m_1} \cdots g_n^{m_n} g^{\mathsf{r}}$$

Pedersen Commitments provide three key properties:

- **Hiding**: The commitment reveals no information about the committed values.
- **Binding**: It's computationally infeasible to open a commitment to different values.
- **Position-Binding**: Each position in the vector is cryptographically bound to its specific base element, preventing attribute swapping.

5.3 We Formally Define a Nullifier

Definition 5.5 (Nullifier Scheme). *A nullifier scheme* $\mathcal{N}$ *is a tuple of algorithms* $(Setup, Gen, Nullify, Verify)$ *where:*

- $Setup(1^\lambda) \to pp$: *Generates public parameters pp based on security parameter* λ.
- $Gen(pp) \to (sk, pk)$: *Generates a secret key sk and public key pk.*

- $Nullify(pp, sk, ctx) \to (nf, \pi)$*: Produces a nullifier* nf *and proof* π *for context* ctx *using secret key* sk*.*
- $Verify(pp, nf, \pi, ctx, S) \to \{0, 1\}$*: Verifies that nullifier* nf *with proof* π *is valid for context* ctx *and that the corresponding public key belongs to a set* S*.*

5.3.1 Security Properties

A secure nullifier scheme satisfies the following properties:

Property 6 (Correctness). For all security parameters λ, contexts ctx, and valid key pairs (sk, pk) generated by Gen:

$$\Pr[Verify(pp, nf, \pi, ctx, S) = 1 \mid (nf, \pi) \leftarrow Nullify(pp, sk, ctx), pk \in S] = 1$$

Property 7 (Uniqueness). It is computationally infeasible to generate two distinct valid nullifiers for the same (sk, ctx) pair. Formally, for any PPT adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{c} Verify(pp, nf, \pi, ctx, S) = 1 \\ \wedge\, Verify(pp, nf', \pi', ctx, S) = 1 \\ \wedge\, nf \neq nf' \end{array} \middle| \begin{array}{c} pp \leftarrow Setup(1^\lambda) \\ (sk, pk) \leftarrow Gen(pp) \\ (nf, \pi, nf', \pi', ctx) \leftarrow \mathcal{A}(pk) \end{array}\right] \leq negl(\lambda)$$

Property 8 (Pseudonymity). Given a valid nullifier nf for context ctx, it is computationally infeasible to determine which public key $pk \in S$ corresponds to nf without knowledge of sk. Formally, for any PPT adversary $\mathcal{A}$:

$$\left|\Pr\left[\mathcal{A}(pp, nf, \pi, ctx, S) = pk \mid \begin{array}{c} (nf, \pi) \leftarrow Nullify(pp, sk, ctx), \\ pk \text{ corresponds to } sk \end{array}\right] - \frac{1}{|S|}\right| \leq negl(\lambda)$$

Property 9 (Indistinguishability). For any PPT adversary $\mathcal{A}$, any contexts ctx_1, ctx_2, and public keys $pk_0, pk_1 \in S$, $\mathcal{A}$ cannot distinguish between nullifiers generated by pk_0 or pk_1:

$$\left|\Pr\left[\mathcal{A}(pp, nf_b, \pi_b, ctx, S) = b \mid \begin{array}{c} b \leftarrow\!\$\, \{0, 1\}, \\ (nf_b, \pi_b) \leftarrow Nullify(pp, sk_b, ctx) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda)$$

where sk_b corresponds to pk_b.

Property 10 (Existential Unforgeability). Without knowledge of a valid secret key sk corresponding to some $pk \in S$, it is computationally infeasible to produce a valid nullifier. Formally, for any PPT adversary $\mathcal{A}$:

$$\Pr\left[Verify(pp, nf, \pi, ctx, S) = 1 \wedge \nexists sk : (sk, pk) \in \mathcal{K} \mid \begin{array}{c} pp \leftarrow Setup(1^\lambda), \\ (nf, \pi, ctx) \leftarrow \mathcal{A}(pp, S) \end{array}\right] \leq negl(\lambda)$$

where $\mathcal{K}$ is the set of all valid key pairs.

5.3.2 Extensions for Probabilistic Nullifiers

For probabilistic nullifiers (committed nullifiers), we extend this definition with:

- $Nullify(pp, sk, ctx, r) \to (cm_{nf}, \pi)$: Produces a commitment to a nullifier cm_{nf} using randomness r, and a proof π.

Property 11 (Unlinkability). For any contexts ctx_1, ctx_2 and any secret key sk, the distributions of nullifier commitments cm_{nf_1} and cm_{nf_2} are computationally indistinguishable, even when generated from the same (sk, ctx) pair with different randomness. Formally, for any PPT adversary $\mathcal{A}$:

$$\left|\Pr\left[\mathcal{A}(pp, cm_{nf_1}, \pi_1, cm_{nf_2}, \pi_2, ctx, S) = 1 \mid \begin{array}{c} r_1, r_2 \leftarrow\!\$\, \mathcal{R}, \\ (cm_{nf_1}, \pi_1) \leftarrow Nullify(pp, sk, ctx, r_1), \\ (cm_{nf_2}, \pi_2) \leftarrow Nullify(pp, sk, ctx, r_2) \end{array}\right]\right.$$

$$- \Pr\left[\mathcal{A}(pp, cm_{nf_1}, \pi_1, cm_{nf_3}, \pi_3, ctx, S) = 1 \;\middle|\; \begin{array}{c} r_1, r_3 \leftarrow\$ \mathcal{R}, \\ (cm_{nf_1}, \pi_1) \leftarrow Nullify(pp, sk, ctx, r_1), \\ (cm_{nf_3}, \pi_3) \leftarrow Nullify(pp, sk', ctx, r_3) \end{array}\right] \leq negl(\lambda)$$

where sk' is a different secret key and $\mathcal{R}$ is the randomness space.

5.4 Efficient Prime-Order Dodis Yampolskiy VRF from the q-DDHI Assumption

The Dodis-Yampolskiy [DY05] VRF provides an elegant solution for generating unique, pseudorandom nullifiers y and their proofs of correctness π. We start with the question of removing pairings to improve efficiency. DY consists of (y, π):

$$y = e(g, \tilde{g})^{1/(sk+x)} \qquad \pi = \tilde{g}^{1/(sk+x)}$$

Verification depends on the bilinearity property $e(g^a, \tilde{g}^b)^c = e(g, \tilde{g})^{abc}$, which enables "exponent multiplication" across groups, allowing the transformations:

$$\begin{aligned} e(g^x \cdot pk, \pi) &\stackrel{?}{=} e(g^{sk+x}, \tilde{g}^{1/(sk+x)}) \\ &\stackrel{?}{=} e(g, \tilde{g})^{(sk+x)/(sk+x)} \\ &= e(g, \tilde{g}) \quad \checkmark \end{aligned} \qquad \begin{aligned} y &\stackrel{?}{=} e(g, \pi) \\ &\stackrel{?}{=} e(g, \tilde{g}^{1/(sk+x)}) \\ &= e(g, \tilde{g})^{1/(sk+x)} \quad \checkmark \end{aligned}$$

Security relies on the q-DBDHI problem which states that given $(g, g^x, g^{x^2}, \ldots, g^{x^q}, \tilde{g})$, no PPT adversary can distinguish between $e(g, \tilde{g})^{1/x}$ and a uniform element in $\mathbb{G}_T$ with non-negligible advantage , ensuring the VRF outputs maintain pseudorandomness after the adversary has observed (x', y', π') pairs.

5.4.1 Technical Challenge

Prime-order groups and standard group operations (without such bilinearity property) cannot be used to directly verify the $1/(sk + x)$ relationship with standard cryptographic operations. For example, given $pk \cdot g^x = g^{sk+x}$, verification attempts to equate or cancel out fail:

1. $g^{sk+x} \cdot g^{1/(sk+x)} = g^{sk+x+\frac{1}{sk+x}}$
2. $\frac{g^{sk+x}}{g^{1/(sk+x)}} = g^{(sk+x)^2 - \frac{1}{sk+x}}$

Our insight is to reverse the verification approach. Instead of trying to derive $g^{1/(sk+x)}$ from g^{sk+x} or cancel with a reciprocal, we use a zero knowledge proof Σ-protocol to verify that y raised to the power $(sk + x)$ equals g. In doing so, our pairing-free construction shifts from the q-DBDHI assumption to the q-DDHI assumption. This gives us the relation:

$$\mathcal{R}_{\mathsf{DY-PF}} = \left\{ \begin{array}{l} (\mathsf{pk}, x, y), \\ (sk) \end{array} \;\middle|\; \begin{array}{l} pk = g^{sk} \\ y^{sk+x} = g \end{array} \right\}$$

5.4.2 Our Sigma Protocol and Construction

Our VRF operates in a prime-order group $\mathbb{G}$ of order p with generator g. The message space is $\mathcal{X} = \mathbb{Z}_p$, the output space is $\mathcal{Y} = \mathbb{G}$, and the proof space is $\Pi = \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p$. The algorithms are defined as follows:

- $\mathsf{VRF.Gen}(1^\lambda) \to (sk, pk)$: Sample $sk \leftarrow\$ \mathbb{Z}_p^*$, compute $pk = g^{sk}$, and output (sk, pk).
- $\mathsf{VRF.Eval}(sk, x) \to y$: Compute $y = g^{1/(sk+x)} \in \mathbb{G}$.

– $\mathsf{VRF.Prove}(sk, x) \to \pi$: Generate proof π using the Σ-protocol described below.

– $\mathsf{VRF.Verify}(pk, x, y, \pi) \to \{0, 1\}$: Output 1 if π verifies y correctly per the Σ-protocol, else 0.

Remark 5.12. Verifying $\mathsf{VRF.Verify}(pk, x, \mathsf{VRF.Eval}(sk, x) \to y) \to 1$ is a naive verification approach without a proof which yeilds a Verifiable Unpredictable Function (VUF), not a VRF because it lacks the mechanism to prove pseudorandomness to a verifier. DY uses pairings to bridge the gap, we replace pairings with a Σ-protocol.

Protocol 2: DY Prime Order Proof Protocol

Common Input: $g, pk, y \in \mathbb{G}$, $x \in \mathbb{Z}_p$

Prover Input: $sk \in \mathbb{Z}_p^*$ with $pk = g^{sk}$, $y = g^{1/(sk+x)}$

Relation:

$$\mathcal{R} = \left\{ (\mathsf{pk}, x, y), (sk) \;\middle|\; pk = g^{sk} \wedge y^{sk+x} = g \right\}$$

1. **Commitment:** Prover samples $r \leftarrow\!\$\, \mathbb{Z}_p$, computes $T_1 = g^r$, $T_2 = y^r$, sends (T_1, T_2).
2. **Challenge:** Verifier samples $c \leftarrow\!\$\, \mathbb{Z}_p$, sends c.
3. **Response:** Prover computes $z = r + c \cdot (sk + x)$, sends z.
4. **Verification:** Verifier checks: $g^z \stackrel{?}{=} T_1 \cdot (pk \cdot g^x)^c$ and $y^z \stackrel{?}{=} T_2 \cdot g^c$

5.4.3 Security Analysis

Our construction replaces the stronger information-theoretic uniqueness of DY VRF with (weaker) computational uniqueness via the Σ-protocol and discrete logarithm assumption.

Correctness Correctness requires that an honest prover's output y and proof π always pass verification. For $pk = g^{sk}$, $y = g^{1/(sk+x)}$, $T_1 = g^r$, $T_2 = y^r$, and $z = r + c(sk + x)$, the verification equations hold:

$$g^z = g^{r+c(sk+x)} = g^r \cdot g^{c(sk+x)} = g^r \cdot (g^{sk} \cdot g^x)^c = T_1 \cdot (pk \cdot g^x)^c$$
$$y^z = y^{r+c(sk+x)} = y^r \cdot y^{c(sk+x)} = y^r \cdot (y^{sk+x})^c = y^r \cdot g^c = T_2 \cdot g^c$$

Since $y^{sk+x} = g^{1/(sk+x)\cdot(sk+x)} = g$, both checks pass, confirming correctness.

Uniqueness Uniqueness ensures that, for a fixed pk and x, only one y can be successfully verified. In DY, pairings enforce this information theoretically. In P-DY, uniqueness is computational, relying on the discrete logarithm problem.

For a valid y, $y^{sk+x} = g$, so $y = g^{1/(sk+x)}$ is unique in $\mathbb{G}$. Suppose an adversary produces $y' \neq y$ with a valid proof π'. Then $y'^{sk+x} = g$ and $y^{sk+x} = g$, implying $(y'/y)^{sk+x} = 1$. In a prime-order group, $y'/y = g^k$ for some $k \neq 0$, so $y' = y \cdot g^k$. But $y'^{sk+x} = (y \cdot g^k)^{sk+x} = g \cdot g^{k(sk+x)} = g$ requires $g^{k(sk+x)} = 1$, which holds only if $k(sk + x) = 0 \pmod p$. For random sk and x, $sk + x = 0$ is negligible. Alternatively, if y' corresponds to a different sk' where $pk = g^{sk'}$, finding $sk' \neq sk$ breaks the discrete logarithm assumption.

Thus, producing a distinct verifiable y' is computationally infeasible, ensuring uniqueness.

Pseudorandomness Pseudorandomness requires that $y = g^{1/(sk+x)}$ appears random in $\mathbb{G}$ without knowledge of sk, even given other input-output pairs. We rely on the q-DDHI assumption, which states that $g^{1/(sk+x)}$ is indistinguishable from random given $(g, g^{sk}, \ldots, g^{(sk)^q})$ for polynomial q. The Σ-protocol is zero-knowledge, leaking no information about sk beyond pk.

Proof (Sketch). Assume an adversary can distinguish y from random, solving q-DDHI. The challenger simulates proofs for q inputs using g^{sk^i} for challenge x^*, provides $y^* = g^{a/(sk+x^*)}$ or a random element. A successful distinguished implies a q-DDHI solver which is assumed to be a hard problem.

5.4.4 Performance Evaluation

We implemented our constructions [Pol25] using the arkworks cryptography library [ark22] in Rust and evaluated performance on a MacBook Air M2 with 16GB RAM. Table 9 compares the baseline Dodis Yampolskiy VRF against our constructions. As our construction does not use bilinear pairings, we present a comparison on BLS12-381 curve as well as more efficient Ed25519 and secp256k1 without a bilinear pairing. Results are available in the figures 14

Table 9. VRF Comparison between Original DY and Our Prime Order DY VRF Construction

Scheme	**Curve**	**Eval + Prove (ms)**	**Verify (ms)**
Our Prime Order DY VRF 5.4.2	Ed25519	0.21	0.37
	secp256k1	0.25	0.42
	BLS12-381	0.41	0.71
Original DY [1] [DY05]	BLS12-381	1.27	2.20

[1]We use optimized pairing for verification by computing all pairings in Miller Loop format before a single Final Exponentiation, reducing verify time from 2.85(ms) to 2.27(ms), a 1.26x speedup.

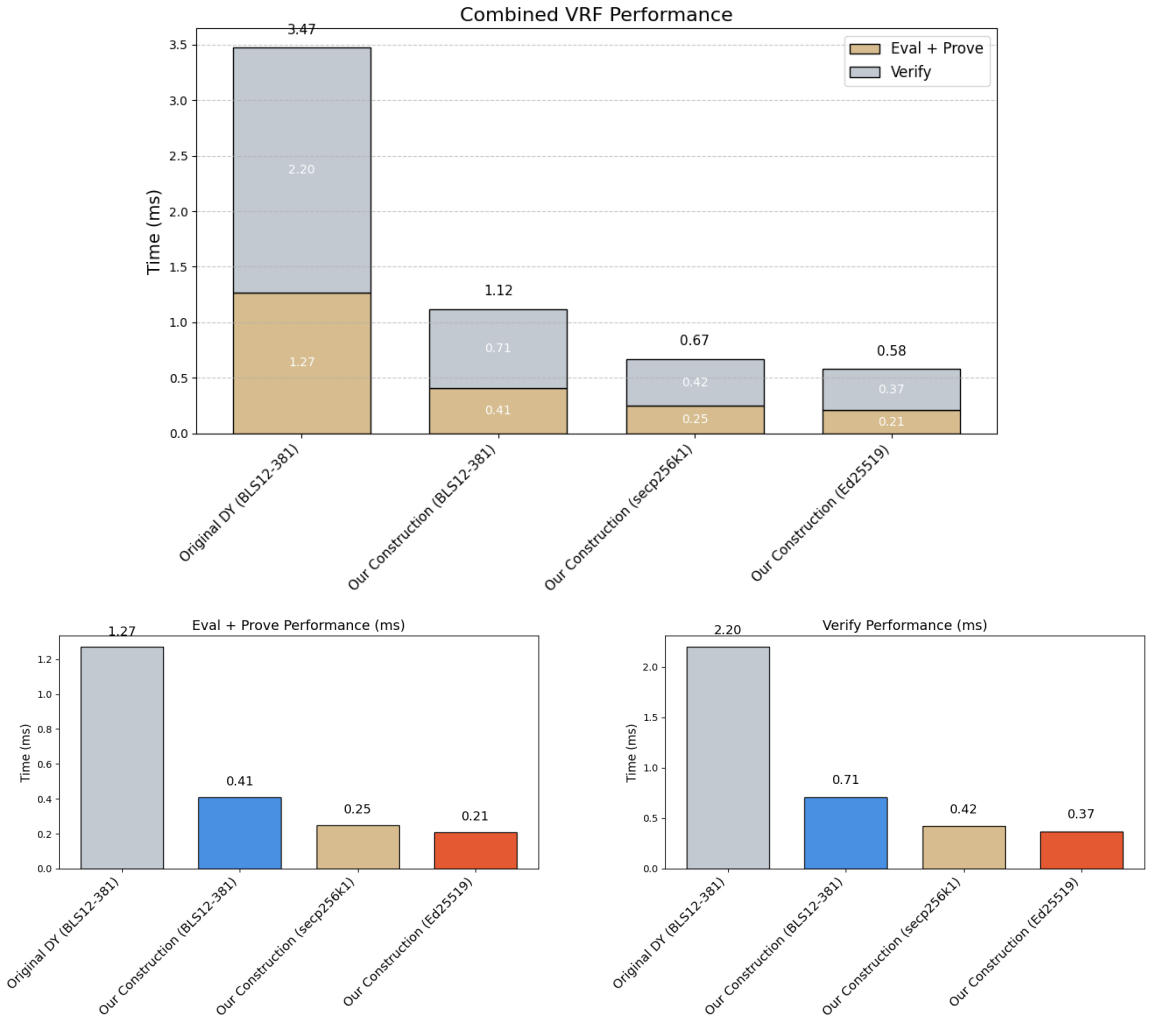

Fig. 13. DY VRF Performance Comparison - Our Construction is 3 - 6x more efficient

5.5 Private Nullifiers from the q-DDHI

While our pairing-free VRF construction, presented in Section 3, eliminates the computational overhead of bilinear pairings, it still exposes the public key $pk = g^{sk}$ and input x during verification. This is a limitation for anonymous credential systems, where both the master credential key sk and the context identifier x must remain private to ensure user anonymity. In this section, we address this challenge by developing zero-knowledge proof techniques that enable verification of VRF outputs computed from committed attributes, without revealing sensitive information.

Our approach builds progressively across three key stages:

1. **Proving Knowledge of a Committed Inverse Exponent**: We start with a foundational protocol to prove that a value $y = g^{1/x}$ corresponds to a committed exponent x, leveraging the q-DDHI assumption. This establishes the base for our next protocols.

2. **Proving Knowledge of a Committed Inverse Linear Relation**: We extend this technique to handle linear combinations of committed values, enabling verification of a VRF output $y = g^{1/(sk+x)}$ where sk and x are committed in separate credentials. This deterministic nullifier supports privacy-preserving credential binding and sybil resistance.

3. **Proving Knowledge of a Committed Nullifier**: Finally, we introduce a protocol to commit to the VRF output itself (i.e., $\mathsf{cm}_y = g_3^{1/(sk+x)} g^{r_3}$), allowing unlinkable presentations. This probabilistic nullifier ensures that repeated uses of the same credential relationship remain unlinkable while still verifiable.

These protocols collectively transform our prime-order VRF into a privacy-preserving tool suitable for hierarchical anonymous credential systems. In the subsections that follow, we detail each stage, presenting the cryptographic constructions and their security properties. **??** will later demonstrate how these building blocks are instantiated for sybil resistance and revocation in the Multi-Issuer Multi-Credential (MIMC) system.

5.5.1 Technical Challenge: Proving Knowledge of the q-DHHI challenge

The central challenge in constructing a pairing-free VRF lies in verifying the correctness of an inverse relationship, specifically that $y = g^{1/x}$ for committed values, without relying on bilinear pairings. In standard prime-order group cryptography, while computing g^x is straightforward, directly verifying $g^{1/x}$ is not, as pairings typically enable this through $e(g^x, g^{1/x}) = e(g, g)$. Our key insight is to reformulate this verification by proving an equivalent relationship: $y^x = g$. This equivalence holds because if $y = g^{1/x}$, then $y^x = (g^{1/x})^x = g^{(1/x)\cdot x} = g$, and conversely, if $y^x = g$, then $y = g^{1/x}$ (assuming $x \neq 0 \mod q$). This transformation eliminates the need for pairings by enabling verification via standard exponentiation, which can be efficiently proven using zero-knowledge techniques.

The protocol we present leverages this equivalence through a zero-knowledge proof where the prover demonstrates knowledge of the opening of a commitment $\mathsf{cm} = g_1^x g^r$ while simultaneously proving that $y^x = g$. This dual verification ensures that y correctly encodes the inverse of x without revealing x, forming a foundation for our pairing-free VRF that balances efficiency and privacy.

5.5.2 Our Sigma-Protocol: Proving Knowledge of a Committed Inverse Exponent

We begin with a simplified relation: given a commitment $\mathsf{cm} = g_1^x g^r$ and an element $y = g^{1/x}$, the prover demonstrates knowledge of x and r satisfying both equations, without revealing them. This is linked to the Decisional Diffie-Hellman Inversion (DDHI) assumption—where distinguishing $g^{1/x}$ from a random element given g, g^x is computationally hard—making it a secure and essential starting point for our privacy-preserving construction.

Protocol 3: Proving Knowledge of Inverse Exponent

Common Input: Group generators $g_1, g \in \mathbb{G}$, commitment $\mathsf{cm} \in \mathbb{G}$, and element $y \in \mathbb{G}$

Prover Input: Witness (x, r) such that $\mathsf{cm} = g_1^x g^r$ and $y = g^{1/x}$

Relation:

$$\mathcal{R}_{\mathsf{DDHI}} = \left\{ (\mathsf{cm}, y), (x, r) \;\middle|\; \mathsf{cm} = g_1^x g^r \;\wedge\; y = g^{1/x} \right\}$$

1. **Commitment:** Prover samples $a_x, a_r \leftarrow\!\!\$\, \mathbb{Z}_q$ and computes:

$$T_1 = g_1^{a_x} g^{a_r}, \quad T_y = y^{a_x}$$

   Sends (T_1, T_y) to the verifier.

2. **Challenge:** Verifier samples $c \leftarrow\!\!\$\, \mathbb{Z}_q$ and sends c to the prover.

3. **Response:** Prover computes:

$$z_x = a_x + c \cdot x, \quad z_r = a_r + c \cdot r$$

   Sends (z_x, z_r) to the verifier.

4. **Verification:** Verifier checks:

$$T_1 \cdot \mathsf{cm}^c \stackrel{?}{=} g_1^{z_x} g^{z_r}, \quad T_y \cdot g^c \stackrel{?}{=} y^{z_x}$$

This protocol achieves two objectives simultaneously:

1. The first verification equation ($T_1 \cdot \mathsf{cm}^c \stackrel{?}{=} g_1^{z_x} g^{z_r}$) proves the prover knows a valid opening (x, r) of the commitment cm. This is a standard Schnorr-type proof for Pedersen commitments.

2. The second equation ($T_y \cdot g^c \stackrel{?}{=} y^{z_x}$) verifies that $y^x = g$, which means $y = g^{1/x}$. This is the critical link proving that y is correctly computed as the inverse exponent.

Security Analysis

Theorem 5.13. *Protocol 5.5.2 is a Σ-protocol for $\mathcal{R}_{\mathsf{DDHI}}$.*

Proof (Sketch). We verify the three properties of a Σ-protocol:

– **Completeness:** For an honest prover with witness (x, r), the verification equations hold:

$$T_1 \cdot \mathsf{cm}^c = g_1^{z_x} g^{z_r} \quad \text{and} \quad T_y \cdot g^c = y^{z_x},$$

as they are satisfied by the protocol's construction.

– **Special Soundness:** Given two accepting transcripts (T_1, T_y, c, z_x, z_r) and $(T_1, T_y, c', z_x', z_r')$ with $c \neq c'$, we can extract:

$$x = \frac{z_x - z_x'}{c - c'} \quad \text{and} \quad r = \frac{z_r - z_r'}{c - c'},$$

such that $\mathsf{cm} = g_1^x g^r$ and $y^x = g$.

– **Honest-Verifier Zero-Knowledge (HVZK):** A simulator samples $z_x, z_r \in \mathbb{Z}_q$ and computes:

$$T_1 = g_1^{z_x} g^{z_r} \cdot \mathsf{cm}^{-c} \quad \text{and} \quad T_y = y^{z_x} \cdot g^{-c},$$

yielding a transcript indistinguishable from a real one.

Relation to q-DDHI Assumption. The q-DDHI assumption ensures that $g^{1/x}$ is computationally indistinguishable from random elements when x is hidden. However, pseudorandomness requires outputs remain indistinguishable even after seeing multiple input-output pairs from the same secret key, and therefore $y = g^{1/x}$ is not pseudorandom.

5.5.3 Our Sigma-Protocol: Proving Knowledge of a Committed Inverse Linear Relation

We now address proving inverse relationships involving linear combinations of committed values. Specifically, we prove knowledge of (sk, x, r_1, r_2) such that:

$$\mathcal{R}_{\mathsf{nullifier}} = \left\{ \begin{pmatrix} (\mathsf{cm}_1, \mathsf{cm}_2, y), \\ (sk, x, r_1, r_2) \end{pmatrix} \;\middle|\; \begin{array}{l} \mathsf{cm}_1 = g_1^{sk} g^{r_1} \\ \mathsf{cm}_2 = g_2^{x} g^{r_2} \\ y = g^{1/(sk+x)} \end{array} \right\}$$

Construction The protocol extends our single-commitment construction by (1) proving knowledge of openings for cm_1 and cm_2 simultaneously, and (2) verifying $y^{sk+x} = g$ where $sk + x$ combines values from distinct commitments. The consistency check $z_m = z_{sk} + z_x$ prevents substitution attacks where an adversary might use different linear combinations across verification equations.

Protocol 4: Proving Committed Inverse Linear Relation

Common Input: Group generators $g_1, g_2, g \in \mathbb{G}$, $\mathsf{cm}_1, \mathsf{cm}_2, y \in \mathbb{G}$

Prover Input: Witness $(sk, x, \mathsf{r}_1, \mathsf{r}_2)$ such that $\mathsf{cm}_1 = g_1^{sk} g^{\mathsf{r}_1}$, $\mathsf{cm}_2 = g_2^{x} g^{\mathsf{r}_2}$ and $y = g^{1/(sk+x)}$

Relation:

$$\mathcal{R} = \{(\mathsf{cm}_1, \mathsf{cm}_2, y), (sk, x, \mathsf{r}_1, \mathsf{r}_2) \mid \mathsf{cm}_1 = g_1^{sk} g^{\mathsf{r}} \wedge \mathsf{cm}_2 = g_2^{x} g^{\mathsf{r}_2} \wedge y = g^{1/(sk+x)}\}$$

1. **Commitment:** Prover computes:

$$a_{sk}, a_x, a_{r_1}, a_{r_2} \leftarrow\!\$\, \mathbb{Z}_q \qquad T_1 \leftarrow g_1^{a_{sk}} g^{a_{r_1}} \qquad T_2 \leftarrow g_2^{a_x} g^{a_{r_2}} \qquad T_y \leftarrow y^{a_{sk}+a_x}$$

   Sends (T_1, T_2, T_y) to verifier.

2. **Challenge:** Verifier samples $c \leftarrow\!\$\, \mathbb{Z}_q$ and sends to prover.

3. **Response:** Prover computes:

$$z_{sk} = a_{sk} + c \cdot sk \qquad z_x = a_x + c \cdot x \qquad z_m = (a_{sk} + a_x) + c \cdot (sk + x)$$

$$z_{r_1} = a_{r_1} + c \cdot r_1 \qquad z_{r_2} = a_{r_2} + c \cdot r_2$$

   Sends $(z_{sk}, z_x, z_{r_1} z_{r_2}, z_m)$ to verifier.

4. **Verification:** Verifier checks:

$$T_1 \cdot \mathsf{cm}_1^c \stackrel{?}{=} g_1^{z_{sk}} g^{z_{r_1}} \qquad T_2 \cdot \mathsf{cm}_2^c \stackrel{?}{=} g_2^{z_x} g^{z_{r_2}} \qquad T_y \cdot g^c \stackrel{?}{=} y^{z_m} \qquad z_m \stackrel{?}{=} z_{sk} + z_x$$

Security Analysis

Theorem 5.14. *Protocol 5.5.3 is a Σ-protocol for $\mathcal{R}_{\mathsf{nullifier}}$.*

Proof (Sketch).

- **Completeness:** For an honest prover, all verification equations hold by similar derivations to the basic case, with the additional consistency check $z_m = z_{sk} + z_x$ holding by construction.
- **Special Soundness:** From accepting transcripts $(T_1, T_2, T_y, c, z_{sk}, z_x, z_{r_1}, z_{r_2}, z_m)$ and $(T_1, T_2, T_y, c', z'_{sk}, z'_x, z'_{r_1}, z'_{r_2}, z'_m)$ with $c \neq c'$, extract $sk = \frac{z_{sk} - z'_{sk}}{c - c'}$, etc. The third verification equation yields $y^{(c-c')(sk+x)} = g^{c-c'}$, thus $y^{sk+x} = g$.
- **HVZK:** Simulator samples $z_{sk}, z_x, z_{r_1}, z_{r_2} \leftarrow\$ \mathbb{Z}_q$, sets $z_m = z_{sk} + z_x$, computes $T_1 = g_1^{z_{sk}} g^{z_{r_1}} \cdot \mathsf{cm}_1^{-c}$, etc.

5.5.4 We Construct a Deterministic Nullifier from 5.5.3

Our VRF operates in a prime-order group $\mathbb{G}$ of order p with generators g_1, g_2, g_3, g. The message space is $x \in \mathcal{X} = \mathbb{Z}_p^*$, the output space is $y \in \mathcal{Y} = \mathbb{Z}_p^*$, and the proof space is $\pi \in \mathit{\Pi} = \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p$. We assume the existence of $\mathsf{cm}_1 = g_1^{sk} g^{r_1}$ and $\mathsf{cm}_2 = g_2^{x} g^{r_2}$.

The algorithms are defined as follows

- $\mathsf{dVRF.Eval}(sk, x) \to y$: $y = g_3^{1/(sk+x)}$
- $\mathsf{dVRF.Prove}(\mathsf{cm}_1, \mathsf{cm}_2, sk, x, y) \to \pi$: Run protocol 5.5.3, output π, cm_y
- $\mathsf{dVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, y, \pi) \to \{0, 1\}$: Output 1 if π verifies $\mathsf{cm}_1, \mathsf{cm}_2, y$ correctly per the Σ-protocol 5.5.3, else 0.

Security Properties

Theorem 5.15 (Correctness). *The deterministic nullifier VRF is correct: for all $\lambda \in \mathbb{N}$, $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{VRF.Gen}(\mathsf{pp})$, $x \in \mathbb{Z}_p$, and $\mathsf{cm}_2 = g_2^x g^{r_2}$:*

$$\Pr[\mathsf{dVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{dVRF.Eval}(\mathsf{sk}, x), \mathsf{dVRF.Prove}(\mathsf{cm}_1, \mathsf{cm}_2, sk, x, y)) = 1] = 1$$

Proof. Correctness follows directly from the completeness of Protocol 5.5.3. Since honest execution ensures all verification equations in the underlying Sigma protocol hold, the VRF verification checks pass.

Theorem 5.16 (Uniqueness). *Under the binding property of Pedersen commitments and the discrete logarithm assumption, for any cm_1 and cm_2, there exists at most one $y \in \mathbb{G}$ such that $\mathsf{dVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, y, \pi) = 1$ for any π.*

Proof (Sketch). The binding property of Pedersen commitments ensures that cm_1 commits to sk and cm_2 commits to x uniquely. Therefore, $y = g^{1/(sk+x)}$ is uniquely determined. If two different values $y' \neq y$ could verify, special soundness would extract witnesses satisfying both $y^{sk+x} = g$ and $y'^{sk+x} = g$, implying $(y'/y)^{sk+x} = 1$. In a prime-order group, this holds only if $y' = y$ or $sk + x \equiv 0 \pmod{p}$. The latter occurs with negligible probability for randomly chosen sk and x.

Theorem 5.17 (Pseudorandomness). *Under the q-DDHI assumption, the deterministic nullifier VRF satisfies pseudorandomness. For any PPT adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{vrf}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{q-DDHI}}(\lambda) + \mathsf{negl}(\lambda)$$

Proof (Sketch). We construct a reduction $\mathcal{B}$ that uses $\mathcal{A}$ to solve q-DDHI:

1. $\mathcal{B}$ receives a q-DDHI challenge $(g, g^a, \ldots, g^{a^q}, Z)$ where Z is either $g^{1/a}$ or random

2. $\mathcal{B}$ embeds a as sk by setting $\mathsf{pk} = \mathsf{cm}_1 = g_1^a \cdot g^{r_1}$

3. For evaluation queries on x_i, $\mathcal{B}$ computes $y_i = g^{1/(a+x_i)}$ using the q-DDHI instance

4. For the challenge x^*, $\mathcal{B}$ returns $y^* = Z^{1/(a+x^*)}$

5. By the HVZK property of the Sigma protocol, $\mathcal{B}$ simulates proofs

6. If $\mathcal{A}$ distinguishes, $\mathcal{B}$ solves the q-DDHI challenge

The advantage is bounded by the q-DDHI advantage plus the negligible HVZK simulation error.

5.5.5 Our Sigma-Protocol: Proving Knowledge of a Committed Inverse Linear Relation

A deterministic nullifier $y = g^{1/(sk+x)}$ is essential for sybil resistance, ensuring a unique, verifiable value for each (sk, x) pair derived from a master credential key sk and context identifier x. However, revealing the nullifier directly introduces linkability risks, as each presentation exposes a deterministic function of (sk, x), potentially allowing user actions to be tracked across interactions. To resolve this, we construct a zero-knowledge Sigma-protocol that proves knowledge of a nullifier embedded within a Pedersen commitment, enabling unlinkable presentations while preserving the verifiable binding necessary for security.

Our solution commits to the nullifier as:

$$\mathsf{cm}_{\text{null}} = g_3^{1/(sk+x)} g^{r_3}$$

where $r_3 \leftarrow\!\!\$\ \mathbb{Z}_q$ provides fresh randomness for each presentation, ensuring unlinkability. We then design a protocol to prove that $\mathsf{cm}_{\text{null}}$ correctly commits to $g_3^{1/(sk+x)}$, where sk and x are themselves committed in $\mathsf{cm}_1 = g_1^{sk} g^{r_1}$ and $\mathsf{cm}_2 = g_2^{x} g^{r_2}$, without revealing any of these values. This approach supports privacy-preserving applications like revocation and set membership proofs in the Multi-Issuer Multi-Credential (MIMC) system.

The key innovation is an auxiliary commitment $\mathsf{cm}_4 = \mathsf{cm}_3^{sk+x} g^{r_4} = g_3 g^{r_3(sk+x)+r_4}$, which allows us to verify that $\mathsf{cm}_3^{sk+x} = g_3$ (modulo randomness) in zero knowledge. This algebraic structure ensures the nullifier's correctness while operating entirely within the commitment space.

Protocol 5: Proof of Committed Nullifier

Public parameters: Prime-order group $\mathbb{G}$ with generators $g_1, g_2, g_3, g \in \mathbb{G}$

Common input: $(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_3, \mathsf{cm}_4) \in \mathbb{G}^4$

Prover input: $(sk, x, r_1, r_2, r_3, r_4) \in \mathbb{Z}_p^6$

Relation:

$$\mathcal{R}_{\mathsf{CN}} = \left\{ \begin{array}{l|l} (\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_3, \mathsf{cm}_4), & \mathsf{cm}_1 = g_1^{sk} g^{r_1} \\ (sk, x, r_1, r_2, r_3, r_4) & \mathsf{cm}_2 = g_2^{x} g^{r_2} \\ & \mathsf{cm}_3 = g_3^{1/(sk+x)} g^{r_3} \quad \text{(committed nullifier)} \\ & \mathsf{cm}_4 = g_3 g^{r_3(sk+x)+r_4} \quad \text{(auxiliary commitment)} \end{array} \right\}$$

1. **Prover** samples $a_{sk}, a_x, a_{r_1}, a_{r_2}, a_{r_3}, a_{r_4} \leftarrow\!\$\, \mathbb{Z}_p$ and computes:

$$T_1 = g_1^{a_{sk}} g^{a_{r_1}} \qquad T_2 = g_2^{a_x} g^{a_{r_2}} \qquad T_3 = g_3^{a_\beta} g^{a_{r_3}} \qquad T_4 = \mathsf{cm}_3^{a_m} g^{a_{r_4}}$$

   where $a_m = a_{sk} + a_x$ and $a_\beta = \frac{1}{a_m}$. Sends (T_1, T_2, T_3, T_4).

2. **Verifier** sends challenge $c \leftarrow\!\$\, \mathbb{Z}_p$.

3. **Prover** computes and sends:

$$z_{sk} = a_{sk} + c \cdot sk \qquad z_x = a_x + c \cdot x \qquad z_{r_i} = a_{r_i} + c \cdot r_i \text{ for } i \in \{1,2,3,4\}$$
$$z_m = a_m + c \cdot (sk + x) \qquad z_\beta = a_\beta + c \cdot \frac{1}{sk + x}$$

4. **Verifier** accepts iff:

$$T_1 \cdot \mathsf{cm}_1^c = g_1^{z_{sk}} g^{z_{r_1}} \qquad T_2 \cdot \mathsf{cm}_2^c = g_2^{z_x} g^{z_{r_2}} \qquad T_3 \cdot \mathsf{cm}_3^c = g_3^{z_\beta} g^{z_{r_3}}$$
$$T_4 \cdot \mathsf{cm}_4^c = \mathsf{cm}_3^{z_m} g^{z_{r_4}} \qquad z_m = z_{sk} + z_x$$

Security Analysis We establish the security of our protocol by proving three core properties: completeness, special soundness, and honest-verifier zero-knowledge (HVZK). These properties ensure that the protocol is correct, extractable, and privacy-preserving, respectively.

Completeness Completeness guarantees that an honest prover, using the correct witness $(sk, x, r_1, r_2, r_3, r_4)$ satisfying the relation $\mathcal{R}_{\mathsf{CN}}$, always convinces the verifier. We verify each step of the protocol to confirm that all verification equations hold.

Assume the prover follows the protocol with:

$$\mathsf{cm}_1 = g_1^{sk} g^{r_1}, \quad \mathsf{cm}_2 = g_2^{x} g^{r_2}, \quad \mathsf{cm}_3 = g_3^{1/(sk+x)} g^{r_3}, \quad \mathsf{cm}_4 = g_3 g^{r_3(sk+x)+r_4}$$

and samples $a_{sk}, a_x, a_{r_1}, a_{r_2}, a_{r_3}, a_{r_4} \leftarrow\!\$\, \mathbb{Z}_p$. Let $a_m = a_{sk} + a_x$ and $a_\beta = \frac{1}{a_m}$, assuming $a_m \neq 0 \pmod p$

The prover computes:

$$T_1 = g_1^{a_{sk}} g^{a_{r_1}}, \quad T_2 = g_2^{a_x} g^{a_{r_2}}, \quad T_3 = g_3^{a_\beta} g^{a_{r_3}}, \quad T_4 = \mathsf{cm}_3^{a_m} g^{a_{r_4}}$$

After receiving challenge c, the prover sends:

$$z_{sk} = a_{sk} + c \cdot sk, \quad z_x = a_x + c \cdot x, \quad z_{r_i} = a_{r_i} + c \cdot r_i \text{ for } i \in \{1,2,3,4\}, \quad z_m = a_m + c \cdot (sk + x), \quad z_\beta = a_\beta + c \cdot \frac{1}{sk + x}$$

The verifier checks:

1. $T_1 \cdot \mathsf{cm}_1^c \stackrel{?}{=} g_1^{z_{sk}} g^{z_{r_1}}$

$$T_1 \cdot \mathsf{cm}_1^c = g_1^{a_{sk}} g^{a_{r_1}} \cdot (g_1^{sk} g^{r_1})^c = g_1^{a_{sk}+c\cdot sk} g^{a_{r_1}+c\cdot r_1} = g_1^{z_{sk}} g^{z_{r_1}}$$

2. $T_2 \cdot \mathsf{cm}_2^c \stackrel{?}{=} g_2^{z_x} g^{z_{r_2}}$

$$T_2 \cdot \mathsf{cm}_2^c = g_2^{a_x} g^{a_{r_2}} \cdot (g_2^{x} g^{r_2})^c = g_2^{a_x+c\cdot x} g^{a_{r_2}+c\cdot r_2} = g_2^{z_x} g^{z_{r_2}}$$

3. $T_3 \cdot \mathsf{cm}_3^c \stackrel{?}{=} g_3^{z_\beta} g^{z_{r_3}}$

$$T_3 \cdot \mathsf{cm}_3^c = g_3^{a_\beta} g^{a_{r_3}} \cdot (g_3^{1/(sk+x)} g^{r_3})^c = g_3^{a_\beta + c/(sk+x)} g^{a_{r_3}+c\cdot r_3} = g_3^{z_\beta} g^{z_{r_3}}$$

4. $T_4 \cdot \mathsf{cm}_4^c \stackrel{?}{=} \mathsf{cm}_3^{z_m} g^{z_{r_4}}$

$$T_4 \cdot \mathsf{cm}_4^c = \mathsf{cm}_3^{a_m} g^{a_{r_4}} \cdot (g_3 g^{r_3(sk+x)+r_4})^c = \mathsf{cm}_3^{a_m} g^{a_{r_4}} \cdot g_3^c g^{c(r_3(sk+x)+r_4)}$$

Since $\mathsf{cm}_3^{sk+x} = g_3 g^{r_3(sk+x)}$ and $\mathsf{cm}_4 = \mathsf{cm}_3^{sk+x} g^{r_4}$, we have:

$$T_4 \cdot \mathsf{cm}_4^c = \mathsf{cm}_3^{a_m} g^{a_{r_4}} \cdot (\mathsf{cm}_3^{sk+x} g^{r_4})^c = \mathsf{cm}_3^{a_m + c(sk+x)} g^{a_{r_4}+cr_4} = \mathsf{cm}_3^{z_m} g^{z_{r_4}}$$

5. $z_m \stackrel{?}{=} z_{sk} + z_x$

$$z_m = a_m + c(sk+x) = (a_{sk} + a_x) + c(sk+x) = z_{sk} + z_x$$

All equations hold, confirming completeness.

Special Soundness Given two accepting transcripts with distinct challenges $c \neq c'$, we extract:

$$sk = \frac{z_{sk} - z'_{sk}}{c - c'}, \quad x = \frac{z_x - z'_x}{c - c'}, \quad r_i = \frac{z_{r_i} - z'_{r_i}}{c - c'} \text{ for } i \in \{1, 2, 3, 4\}$$

From $z_\beta = a_\beta + c \cdot \frac{1}{sk+x}$ and $z'_\beta = a_\beta + c' \cdot \frac{1}{sk+x}$, we derive $\frac{1}{sk+x} = \frac{z_\beta - z'_\beta}{c-c'}$, ensuring cm_3 commits to $g_3^{1/(sk+x)} g^{r_3}$. This confirms extractability.

Honest-Verifier Zero-Knowledge (HVZK) A simulator samples $z_{sk}, z_x, z_{r_1}, z_{r_2}, z_{r_3}, z_{r_4}, z_m, z_\beta \leftarrow\!\$\ \mathbb{Z}_p$ and computes:

$$T_1 = g_1^{z_{sk}} g^{z_{r_1}} \cdot \mathsf{cm}_1^{-c}, \quad T_2 = g_2^{z_x} g^{z_{r_2}} \cdot \mathsf{cm}_2^{-c}, \quad T_3 = g_3^{z_\beta} g^{z_{r_3}} \cdot \mathsf{cm}_3^{-c}, \quad T_4 = \mathsf{cm}_3^{z_m} g^{z_{r_4}} \cdot \mathsf{cm}_4^{-c}$$

These satisfy the verification equations, and the simulated transcript's distribution matches the real one, ensuring zero-knowledge.

5.5.6 We Construct a Probabilistic Nullifier from 5.5.5

Algorithm Syntax Here, the output is a commitment $\mathsf{cm}_y = g_3^{1/(sk+x)} g^{r_3}$, randomized per presentation, with an auxiliary commitment cm_4 for verification.

- $\mathsf{pVRF.Eval}(sk, x) \to \mathsf{cm}_y$: Sample $r_3 \leftarrow\!\$\ \mathbb{Z}_p^*$, compute $\mathsf{cm}_y = g_3^{1/(sk+x)} g^{r_3}$, sample $r_4 \leftarrow\!\$\ \mathbb{Z}_p^*$ compute $\mathsf{cm}_4 = g_3 g^{r_3 \cdot (sk+x)+r_4}$ return $\mathsf{cm}_y, \mathsf{cm}_4$
- $\mathsf{pVRF.Prove}(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4, sk, x, r_1, \ldots, r_4) \to \pi$: Run protocol 5.5.3, output π
- $\mathsf{pVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4, \pi) \to \{0, 1\}$: Output 1 if π verifies $\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4$ correctly per the Σ-protocol 5.5.5, else 0.

Security Properties

Theorem 5.18 (Correctness). *The probabilistic nullifier VRF is correct following from the completeness of Protocol 5.5.5.*

Theorem 5.19 (Uniqueness). *For fixed* pk *and* cm_2*, the underlying nullifier value* $g^{1/(sk+x)}$ *is unique, though committed differently each time. The binding property of commitments ensures* sk *and* x *are uniquely determined.*

Theorem 5.20 (Unlinkability). *For any PPT adversary* $\mathcal{A}$ *making polynomially many queries, the following experiment has negligible advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{unlink}}(\lambda) = \left|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink}-1}(\lambda) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{unlink}-0}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{unlink}-\mathsf{b}}$ *is:*

1. $\mathcal{A}$ *chooses* $(\mathsf{pk}, \mathsf{cm}_{2,0}, \mathsf{cm}_{2,1})$ *where* $\mathsf{cm}_{2,0}, \mathsf{cm}_{2,1}$ *commit to the same* x
2. *Challenger computes* $(\mathsf{cm}_{y,b}, \pi_b) \leftarrow \mathsf{VRF.Eval}(\mathsf{sk}, x), \mathsf{VRF.Prove}(\mathsf{sk}, x, \mathsf{cm}_{2,b})$
3. $\mathcal{A}$ *outputs* b' *given* $(\mathsf{cm}_{y,b}, \pi_b)$

Proof (Sketch). Unlinkability follows from:

1. The hiding property of Pedersen commitments with fresh randomness r_3, r_4
2. The HVZK property of Protocol 5.5.5, which reveals nothing beyond statement validity
3. The statistical independence of commitments across presentations

Each presentation uses independent randomness, making transcripts indistinguishable even for the same underlying (sk, x) pair.

5.6 Performance Evaluation

We implemented our constructions [Pol25] using the arkworks cryptography library [ark22] in Rust and evaluated performance on a MacBook Air M2 with 16GB RAM. Table 10 compares our constructions on BLS12-381 curve against baseline approaches, focusing on the critical operations of evaluation, proof generation, and verification. Table 11 shows our constructions built on different, non-pairing curves.

Table 10. VRF Comparison with curve BLS12-381

Scheme	**Pairing**	**Eval + Prove (ms)**		**Verify (ms)**	
		ms	**Speedup**	**ms**	**Speedup**
DY [1] [DY05]	yes	1.27	-	2.2	-
Us (Prime Order DY) 5.4.2	no	0.41	3.1x	0.70	3.1x
DY Anonymous [TBA+22]	yes	5.91	-	6.47	-
Us (Anonymous + Deterministic) 5.5.3	no	1.08	5.5x	1.41	4.6x
Us (Anonymous + Probabilistic) 5.5.5	no	2.01	2.9x	1.65	3.9x

[1]We use optimized pairing for verification by computing all pairings in Miller Loop format before a single Final Exponentiation, reducing verify time from 2.85(ms) to 2.27(ms), a 1.26x speedup.

Our evaluation reveals several key findings:

Table 11. Performance of our VRF Constructions Across Elliptic Curves (time in ms)

Curve	Scheme	Eval + Prove		Verify	
		ms	Speedup	ms	Speedup
BLS12-381	Prime Order DY 5.4.2	0.41	-	0.71	-
	Prime Order, Anon. Deterministic 5.5.3	1.15	-	1.08	-
	Prime Order, Anon. Probabilistic 5.5.5	2.01	-	1.65	-
Ed25519	Prime Order DY 5.4.2	0.21	2.0×	0.37	1.9×
	Prime Order, Anon. Deterministic 5.5.3	0.56	2.0×	0.56	1.9×
	Prime Order, Anon. Probabilistic 5.5.5	1.04	1.9×	0.84	2.0×
secp256k1	Prime Order DY 5.4.2	0.25	1.6×	0.42	1.7×
	Prime Order, Anon. Deterministic 5.5.3	0.66	1.8×	0.66	1.6×
	Prime Order, Anon. Probabilistic 5.5.5	1.29	1.6×	1.05	1.6×

Measurements conducted on a MacBook Air M2 with 16GB RAM using the arkworks library [ark22] in Rust. Speedup is calculated relative to BLS12-381 for each scheme and operation. Times are rounded to two decimal places for clarity.

1. **Pairing Elimination:** Our DY-PF construction achieves a 3.1× speedup over the original DY VRF by eliminating pairing operations, demonstrating the efficiency benefits of our approach.

2. **Privacy Preservation Overhead:** Adding privacy preservation through commitments inevitably increases computational costs, but our DY-PF-Private design achieves a 5.5× speedup over comparable pairing-based private VRF schemes.

3. **Unlinkability Trade-off:** The committed nullifier variant (DY-PF-Private-ComOut) involves additional operations for commitment handling, yet still maintains a 2.9× evaluation speedup and 3.9× verification speedup over pairing-based alternatives.

4. **Inefficiency of Pairing Curve Operations** The non-pairing curve operations are from 1.6x - 2x more efficient

5.7 Instantiation: Anonymous Nullifiers for Sybil Resistance and Revocation

In this section, we instantiate the Credential Relationship Binding Nullifier (CRBN) within the Multi-Issuer Multi-Credential (MIMC) system, demonstrating its utility in real-world privacy-preserving identity frameworks. The CRBN leverages the prime-order, pairing-free Verifiable Random Function (VRF) constructions from Sections 5.4 and 5.5 to provide efficient, anonymous nullifiers. We focus on two primary applications—sybil resistance and revocation—while highlighting additional use cases inspired by novel zkp applications.

5.7.1 Applications

The CRBN enables a range of privacy-preserving applications by binding credentials hierarchically while enforcing uniqueness and revocability. Below, we outline key use cases:

Sybil Resistance Sybil resistance ensures that users cannot create multiple identities to exploit a system. The CRBN achieves this through its deterministic nullifier, $y = g^{1/(sk+x)}$, where sk is the master credential key and x is the context identifier (e.g., "DMV" for a driver's license). This nullifier is unique per user-context pair and verifiable via the zero-knowledge proof from Protocol 5.5.3.

Example: Anonymous Voting. Consider a voting system where each user should vote once per election. A user with master credential key sk computes $y = g^{1/(sk+x)}$ for election context x (e.g., "Election2024"). They present y with a zero-knowledge proof that it corresponds to a valid credential, without revealing sk or x. The system maintains an anonymous user list (see Figure 11) and rejects duplicate nullifiers, preventing double voting while preserving anonymity.

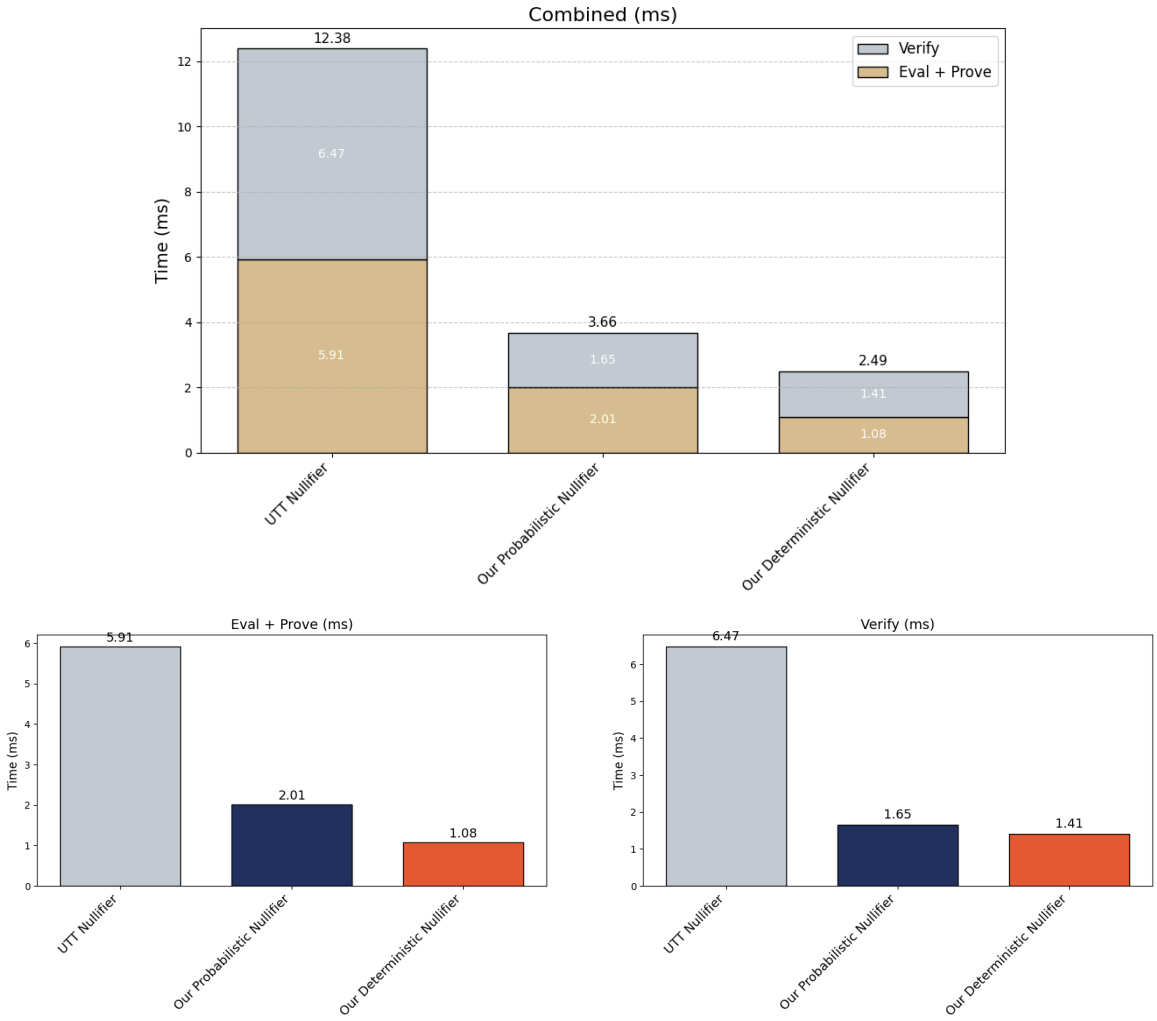

Fig. 14. Nullifier Performance Comparison - Our Constructions are 3x - 5x more efficient

Revocation Revocation allows credentials to be invalidated without compromising privacy. Suppose an identity system has some connection between a user's nullifier, which was submitted at the beginning of their relationship with the system and their user. Should the system require to revoke the user, they place the nullifier in the revocation list. The committed CRBN variant, $\mathsf{cm}_y = g_3^{1/(sk+x)} g^{r_3}$ (Protocol 5.5.5), supports this by embedding the nullifier in a commitment, enabling unlinkable presentations and privacy-preserving revocation checks.

Example: Service Access. A user accesses a service with a credential tied to context x (e.g., "ServiceA"). If revocation is required, the service adds the nullifier $g^{1/(sk+x)}$ to a revocation list. Subsequent access attempts require a zero-knowledge proof that the user's nullifier is not in the list, using techniques like set membership proofs [CL02]. The committed variant ensures each presentation is unlinkable, protecting user privacy.

Additional Use Cases Beyond sybil resistance and revocation, the CRBN supports diverse applications, drawing from recent privacy-preserving protocol designs:

- **ZK Airdrops:** In cryptocurrency airdrops, users prove eligibility (e.g., ownership of a key in a Merkle tree) and uniqueness via a nullifier, claiming rewards once anonymously.
- **Persistent Anonymous Identities:** On message boards, users post under a consistent nullifier (e.g., $y = g^{1/(sk+\text{"threadID"})}$), linking posts to one identity without revealing it.

– **Uniqueness Claims:** For proof of ownership (e.g., a unique asset), a nullifier ensures single claims while maintaining anonymity.

5.7.2 Discussion

The CRBN's pairing-free design and zero-knowledge proofs provide efficiency and privacy advantages over pairing-based alternatives (e.g., UTT [TBA+22]), as shown in Section 5.6. Its flexibility supports both deterministic (sybil resistance) and committed (revocation) variants, addressing diverse needs in anonymous credential systems.

5.7.3 Conclusion

The CRBN instantiates a versatile, efficient nullifier mechanism for the MIMC system, enabling sybil-resistant credential binding and privacy-preserving revocation. Its applications extend to emerging use cases like ZK airdrops and persistent identities, positioning it as a foundational tool for next-generation privacy-preserving identity frameworks.

Chapter 6

Sybil Resistant Threshold Issued Identity System from Anonymous Credentials and Anonymous Nullifiers

6.1 Introduction

Self-sovereign identity (SSI) systems enable users to control their digital identities with strong privacy, Sybil resistance, and minimal issuer interaction. Distributing trust across multiple issuers is critical to avoid single points of failure. CanDID [MMZ+20] is a pioneering SSI system that uses multi-party computation (MPC) to deduplicate identities and issue credentials via a (threshold) committee of signing nodes. While effective, it has drawbacks: slow MPC-based deduplication, frequent issuer interactions, linkability risks, and little protection for credential non-transferability. Similar to Coconut [SABB+20], a threshold credential system that distributes trust across multiple issuers, T-SIRIS employs a t-out-of-N threshold issuance model to enhance decentralization. However, Coconut lacks a mechanism to prevent sybil attacks, a critical requirement for self-sovereign identity (SSI) systems. T-SIRIS addresses this gap by integrating anonymous, deterministic nullifiers from our Credential Relationship Binding Nullifier (CRBN) scheme (Chapter 5), adding just 2.49ms to issuance overhead to credential issuance (Section 6.5.2) while ensuring users cannot obtain multiple credentials per context.

This chapter presents T-SIRIS, a threshold Sybil-resistant identity system built on our multi-issuer multi-credential anonymous credential system (MIMC-ABC) from Chapter 4 and Credential Relationship Binding Nullifier (CRBN) from Chapter 5. T-SIRIS uses a t-out-of-N threshold issuance model, secure against $t-1$ malicious issuers. We compare T-SIRIS with CanDID and S3ID [RAR+24], which uses threshold anonymous counting tokens (tACT), showing improvements in efficiency and functionality.

Chapter Roadmap

The remainder of this chapter is structured as follows: Section 6.2 covers cryptographic preliminaries. Section 6.3 presents the T-SIRIS system model and security requirements. Section 6.4 details our construction and protocols. Section 6.5 evaluates our performance against state-of-the-art alternatives.

6.1.1 Motivation and Challenges

Traditional Attribute-Based Anonymous Credential systems (ABCs) are based on a centralized issuer, which has its own security flaws - a malicious issuer can issue credentials that can't be traced in an anonymous system, undermining the system's integrity. To address this, we distribute security among N nodes, requiring a threshold of at least t nodes to issue a valid credential.

CanDID [MMZ+20] pioneered Sybil-resistant identity with legacy credential oracles, but relies on costly multi-party computation (MPC) among committee nodes for deduplication and requires interactive issuance for context credentials, contradicting self-sovereign principles. Each transaction requires communication with committee members, introducing privacy vulnerabilities where a single malicious node can link different user transactions.

CanDID CanDID deduplicates a user attribute, e.g. social security number using MPC, storing a unique value in a table T_{Dedup}. Users create a master credential from legacy data via oracles, then request context-specific credentials. Its limitations include:

- **Deduplication:** MPC deduplication is slow and reveals a pseudonym to issuers, enabling pseudonym linking. T-SIRIS uses CRBN, computing a nullifier $y = g^{1/(s+\text{ctx})} \in \mathbb{G}_1$ with user key $k \in \mathbb{Z}_p$ and context ctx. This is anonymous and many times faster than tACT [RAR+24] (see Section **??**).
- **Efficient Deduplication and Verification:** T-SIRIS achieves fast deduplication using CRBN, computing a nullifier $y = g^{1/(s+\text{ctx})} \in \mathbb{G}_1$ with user key $k \in \mathbb{Z}_p$ and context ctx. Unlike Coconut [SABB+20], which lacks sybil resistance, and CanDID's MPC-based deduplication, CRBN is pairing-free and 5× faster than S3ID's tACT [RAR+24]. Verification scales sublinearly with attributes n, achieving up to 44.1× speedup over tACT for $n = 64$ (see Section **??**).
- **Context Credential Issuance:** CanDID requires issuer interaction for each context credential, e.g., "over 18." T-SIRIS users receive credentials from different issuers and use ZKPs to prove predicates $\phi(\vec{m}) = m_1 \geq 18$ based on the attributes in their set of credentials, allowing flexibility and dynamic verification scenarios without interacting with an issuer each time.
- **Sybil Resistance:** CanDID tracks public keys, risking linkability. T-SIRIS embeds s in all credentials, using CRBN with ZKP $\Pi^{\mathcal{R}_{\text{null}}} = \text{ZKPoK}\{(k) : y = g^{1/(s+\text{ctx})}\}$ for unlinkable Sybil resistance.

Both systems use a master-context credential hierarchy, but T-SIRIS allows different issuers for context credentials and supports flexible ZKP-based predicates.

S3ID/tACT [RAR+24] uses tACT, based on Shacham-Waters signatures, for Sybil resistance and threshold issuance. Both T-SIRIS and S3ID embed a user key s in credentials. T-SIRIS improves in:

- **Efficiency:** Our verification is nearly constant as the number of attributes in signatures increases due to efficient Schnorr signatures and Multi-Scalar Multiplication techniques, offering up to 44.1× faster verification for $n = 64$ attributes (see Section **??**).
- **Predicate Support:** S3ID's Groth-Sahai proofs limit predicate expressiveness or trade expressiveness for efficiency. T-SIRIS's MIMC-ABC supports complex computation over group exponents with efficient Schnorr proofs.

Managing SSI Features with Privacy Preserving Cryptography SSI requires unlinkability, Sybil resistance, and efficiency. Approaches include tokens (tACT [RAR+24]), zkSNARKs [RWGM22], and anonymous credentials (ABC). T-SIRIS's ABC-based design with CRBN and MIMC-ABC balances speed and flexibility, scaling well with attributes n and issuers N.

6.1.2 Contributions

Our Threshold Sybil-Resistant Identity System (T-SIRIS) advances self-sovereign identity (SSI) by combining efficiency, expressiveness, and security in a multi-issuer setting. Built on the Multi-Issuer Multi-Credential Anonymous Credential system (MIMC-ABC) from Chapter 4 and the Credential Relationship Binding Nullifier (CRBN) from Chapter 5, T-SIRIS outperforms existing systems like CanDID [MMZ+20] and S3ID [RAR+24]. Below, we outline our contributions, aligning them with S3ID's claims to highlight T-SIRIS's improvements.

1. **Efficient Deduplication and Verification:** T-SIRIS achieves fast deduplication using CRBN, computing a nullifier $y = g^{1/(s+\text{ctx})} \in \mathbb{G}_1$ with user key $s \in \mathbb{Z}_p$ and context ctx. Unlike Coconut [SABB+20], which lacks sybil resistance, and CanDID's MPC-based deduplication, CRBN is pairing-free and many times faster than S3ID's tACT [RAR+24]. Verification scales sublinearly with attributes n, achieving up to 44.1× speedup over tACT for $n = 64$

2. **Expressive Predicate Proofs:** T-SIRIS supports complex predicates (e.g., $\phi(\vec{m}) = m_1 \geq 18 \wedge m_2 \in S$) via MIMC-ABC and Schnorr-based zero-knowledge proofs (ZKPs). Unlike S3ID's Groth-Sahai proofs, which limit expressiveness, T-SIRIS enables range proofs and set membership with sublinear complexity, enhancing flexibility for SSI applications (see Chapter 6).

3. **Provably Secure SSI:** T-SIRIS provides formal security definitions and proofs for unforgeability, Sybil resistance, strong unlinkability, and identity binding, secure against $t-1$ malicious issuers in a t-out-of-N threshold model. Our proofs, based on the q-DDHI assumption for CRBN and SDLP for MIMC-ABC, extend S3ID's tACT-based security to multi-issuer settings with identity binding (see Section 4).

4. **Non-transferability and Identity Binding:** T-SIRIS embeds a user identifier id in all credentials, ensuring only the owner can use them, similar to S3ID.

Comparison with S3ID/tACT: S3ID [RAR+24] claims efficient deduplication, non-interactive credential generation, unlinkability, and non-transferability using tACT. T-SIRIS achieves these with superior efficiency and expressiveness (Schnorr vs. Groth-Sahai). While S3ID supports single-issuer deduplication, T-SIRIS's MIMC-ABC enables multi-issuer identity binding, addressing a broader range of SSI use cases. Both systems ensure non-transferability via a private key, but T-SIRIS's ZKP $\Pi^{\mathcal{R}_{\text{null}}} = \text{ZKPoK}\{(k) : y = g^{1/(s+\text{ctx})}\}$ offers stronger unlinkability against colluding issuers.

Table 12. Comparison of T-SIRIS with prior SSI systems.

System	Sybil Resist.	Threshold	Non-Interact.†	Non-Transfer.‡	Complex Pred.§	M.I. Anon.¶
CanDID [MMZ+20]	✓[a]	✓	✗	✗	✗	✗
SyRA [CKS24]	✓	✗	✓	✗	✗	✗
S3ID [RAR+24]	✓	✓	✓	✓	✗	✗
Coconut [SABB+20]	✗	✓	✓	✗	✓	✗
T-SIRIS (Ours)	✓	✓	✓	✓	✓	✓

† Non-interactive application credential generation (Section **??**).
‡ Credentials bound to the true owner (Section **??**).
§ Support for complex predicates, e.g., range proofs (Section **??**).
¶ Anonymity against malicious issuers (Section **??**).
[a] Pseudonymous, not fully anonymous.
Note: We compare with CanDID for its SSI prominence, SyRA for its VRF-based Sybil resistance, S3ID for its threshold similarity, and Coconut for its threshold issuance baseline.

6.2 Preliminaries

Definition 6.1 (Shamir Secret Sharing). *A (t, n) secret sharing scheme* SS *consists of the following* PPT *algorithms over message space $\mathcal{X}$:*

- $\mathsf{Share}(1^\lambda, t, n, x) \to ([x]_1, \ldots, [x]_n)$: *Takes security parameter λ, threshold t, number of parties n, and secret $x \in X$. Outputs n shares $([x]_1, \ldots, [x]_n)$.*

- $\mathsf{Combine}([x]_{i_1}, \ldots, [x]_{i_t}) \to x$: *Takes as input t distinct shares $[x]_{i_j}$ where $i_j \in [n]$ for $j \in [t]$, and outputs the reconstructed secret $x \in \mathcal{X}$.*

6.2.1 Threshold Rerandomizable Signature

Our base-building block is the threshold PS signature scheme from [TBA+22] using Shamir secret sharing with similarities to [SABB+20] to distribute trust among multiple issuers. The key pair $(\mathsf{sk}, \mathsf{vk})$ is (t, n)-secret-shared among the signers, where each signer i holds a share secret key sk_i and a share verification key vk_i. This enables subsets of at least $t+1$ signers to collaboratively produce a signature that verifies under the combined verification key vk, while ensuring that no subset of t or fewer signers can forge signatures.

We assume up to $t-1$ malicious issuers in an n-issuer system, where t is the threshold, reflecting realistic collusion scenarios in decentralized settings. The adversary cannot break the hard problems our schemes are built upon, and network conditions support standard threshold protocols. Similar to Coconut's threshold issuance [SABB+20], our threshold PS signature scheme distributes trust among n issuers using Shamir secret sharing. However, we extend this foundation by integrating nullifiers from Chapter 5, enabling sybil resistance—a critical feature for identity systems that Coconut does not address.

Importantly, this threshold construction distributed trust among multiple issuers while preserving the core properties of our underlying building block PS signature scheme - unforgeability and rerandomizability. Additionally, the end-signature algebraic structure is the same, and therefore, the verification algorithm and proof system compatibility are the same.

Definition 6.2 (PS Threshold Signatures over Pedersen Commitments). *A PS threshold signature scheme* RS *over Pedersen commitments consists of the following algorithms:*

- $\mathsf{DistKeyGen}(1^\lambda, t{+}1, n, \ell) \to (\mathsf{ck}, \mathsf{vk}, (\mathsf{sk}_i, \mathsf{vk}_i)_{i\in[n]})$: *Takes security parameter λ, corruption threshold t, number of parties n, and attribute count ℓ. Outputs commitment key* ck*, verification key* vk*, and per-party keys* $(\mathsf{sk}_i, \mathsf{vk}_i)$*.*

- $\mathsf{ShareSign}(\mathsf{ck}, \mathsf{sk}_i, (\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}})_{k\in[\ell]}; h) \to [\sigma^*]_i$: *Takes party i's secret key* sk_i*, commitments* cm_k *with ZK proofs* π_k^{zkpok}*, and randomizer h. Outputs signature share* $[\sigma^*]_i$*.*

- $\mathsf{ShareVer}(\mathsf{ck}, \mathsf{vk}_i, (\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}})_{k\in[\ell]}, [\sigma^*]_i; h) \to \{0,1\}$: *Takes party i's verification key* vk_i*, commitments with proofs, and signature share* $[\sigma^*]_i$*. Outputs accept (1) or reject (0).*

- $\mathsf{Aggregate}(\mathsf{ck}, \{[\sigma^*]_i\}_{i\in S}, \{r_k\}_{k\in[\ell]}) \to \sigma$: *Takes signature shares from subset S of parties and commitment randomness values. Outputs aggregated signature σ.*

- $\mathsf{Rerand}(\mathsf{vk}, \sigma, \Delta_r, \Delta_u) \to \sigma'$*: Creates a rerandomized signature σ' from signature σ using randomization values Δ_r, Δ_u.*

- $\mathsf{Verify}(\mathsf{vk}, \mathsf{cm}, \sigma) \to \{0,1\}$*: Verifies signature σ on commitment* cm *using public key* vk*.*

Construction Our threshold PS construction works as follows: A user with attributes $\mathsf{attrs} = [m_1, \ldots, m_\ell]$ generates a commitment $\mathsf{cm} = \mathsf{CM.Com}([\mathsf{attrs}]; r)$. The user interacts with signers using a pre-agreed random element $h \in \mathbb{G}_1$, alternatively, [SABB+20] generates this with a hash-to-group of the commitment $h \leftarrow \mathcal{H}(\mathsf{cm})$. The user sends cm with $\Pi^{\mathcal{R}_{\mathsf{com}}}$ proving its opening to each signer. Signers verify the proof and return a signature share $[\sigma^*]_i = \mathsf{tPSutt.ShareSign}(\mathsf{ck}, \mathsf{sk}_i, (\mathsf{cm}_k, \Pi_k^{\mathcal{R}_{\mathsf{com}}}); h)$. The user verifies each share and checks the consistency of the message and proof previously shared $\mathsf{tPSutt.ShareVer}(\mathsf{ck}, \mathsf{vk}_i, (\mathsf{cm}, \pi^{\mathsf{zkpok}}), [\sigma^*]_i; h)$. The user identifies a set S of at least $t+1$ valid shares and aggregates them, outputting their signature $\sigma \leftarrow \mathsf{tPSutt.Aggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, r)$ which verifies under the combined verification key $\mathsf{PSutt.Ver}(\mathsf{vk}, \mathsf{cm}, \sigma) = 1$.

- $\mathsf{tDistKeyGen}(1^\lambda, t+1, n, \ell) \to (\mathsf{ck}, \mathsf{vk}, h, (\mathsf{sk}_i, \mathsf{vk}_i)_{i\in[n]})$: Takes input the security parameter, t the corruption threshold, n is number of nodes, ℓ is the credential message length. Outputs ck, vk the commitment and verification keys generated with the secrets and $\mathsf{sk}_i, \mathsf{vk}_i$ the shared keys to distribute to n nodes. let $\mathcal{X}$ and ψ_k be t degree polynomials:

- For the shared x value: $\mathcal{X} \leftarrow\!\!\$\ \mathbb{Z}_p[X], x \leftarrow \mathcal{X}(0), \{[x]_i \leftarrow \mathcal{X}(i)\}_{i\in[n]}$
- For the shared u value: $\mu \leftarrow\!\!\$\ \mathbb{Z}_p[X], h \leftarrow \mu(0), \{[h]_i \leftarrow \mu(i)\}_{i\in[n]}$
- For the s shared y values: $\{\psi_k \leftarrow\!\!\$\ \mathbb{Z}_p[X], y_k \leftarrow \psi_k(0)\}_{k\in[\ell]}, \{[y_k]_i \leftarrow \psi_k(i)\}_{k\in[\ell],i\in[n]}$
- using the secret, non-shared values, trusted setup party computes $\mathsf{vk} \leftarrow \tilde{g}^x, h \leftarrow g^u, \mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda, \ell, \vec{y})$ and parses ck as $(g, \vec{g}, \tilde{g}, \vec{\tilde{g}})$. Note that $g_k = g^{y_k}, \tilde{g}_k = \tilde{g}^{y_k}$
- computes shared secret values $\{\mathsf{sk}_i \leftarrow ([x]_i, ([y_k]_i))_{k\in[\ell]}\}_{i\in[n]}, \{\mathsf{vk} \leftarrow (\tilde{g}^{[x]_i}, (\tilde{g}^{[y_k]_i})_{k\in[\ell]})\}_{i\in[n]}$
- Return $\mathsf{ck}, \mathsf{vk}, \{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[n]}$

– $\mathsf{tPrepare}(\mathsf{ck}, h, \mathsf{attrs}) \rightarrow (\{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h)$: User commits to each m individually:

- Samples $\{r_k\}_{k\in|\ell|} \leftarrow\!\!\$\ \mathbb{Z}_p^*$
- Computes $\mathsf{cm}_k = h^{m_k} g^{r_k}$ and generates proofs for each $\{\mathsf{cm}_k = \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in|\ell|}$

– $\mathsf{tShareSign}(\mathsf{ck}, \mathsf{sk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, h) \rightarrow [\sigma^*]_i$: the user runs $\mathsf{tShareSign}$ with at least t nodes.

- Signer parses g from ck and verifies $\{\mathsf{ZK.Verify}(\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k))\}_{k\in[\ell]}$
- Signer parses sk_i as $[x]_i, \{[y_k]_i\}_{k\in[\ell]}$
- Signer signs their share $[\sigma^*]_i \leftarrow (h, h^{[x]_i} \prod_{k\in[\ell]} \mathsf{cm}_k^{[y_k]_i}) = (h, h^{[x]_i + \sum_{k\in[\ell]} m_k [y_k]_i} \cdot g^{\sum_{k\in[\ell]} r_k [y_k]_i})$

– $\mathsf{tShareVer}(\mathsf{ck}, \mathsf{vk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, [\sigma^*]_i; h) \rightarrow \{0,1\}$: Run by a user to verify the signed share returned from each node before aggregating together.

- Parse $[\sigma^*]_i$ as $([\sigma^*]_{i,1}, [\sigma^*]_{i,2})$ and check $h = [\sigma^*]_{i,1}$.
- Verifies $\{\mathsf{ZK.Verify}(\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k))\}_{k\in[\ell]}$
- Parses vk_i as $(\tilde{g}^{[x]_i}, \{\tilde{g}^{[y_k]_i}\}_{k\in[\ell]})$
- Asserts $e([\sigma^*]_{i,2}, \tilde{g}) = e(h, \tilde{g}^{[x]_i}) \cdot \prod_{k\in[\ell]} e(\mathsf{cm}_k, \tilde{g}^{[y_k]_i})$

– $\mathsf{tAggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, \{r_k\}_{k\in[\ell]}) \rightarrow \sigma$: User parses ck as $(\cdot, \vec{g}, \cdot, \cdot)$ and $\forall i \in S$, parse $[\sigma^*]_i$ as $(h, [\sigma^*]_{i,2})$. Runs Lagrange Interpolation on $|S| = t+1$ signature shares:

- $\mathcal{L}_i \leftarrow \prod_{j\in S, j\neq i} \frac{0-j}{i-j} \forall i \in S$
- $\sigma_2 \leftarrow \prod_{j\in S} ([\sigma^*]_{i,2})^{\mathcal{L}_i} = (h, h^{x + \sum_{k\in[\ell]} m_k y_k} \cdot g^{\sum_{k\in[\ell]} r_k y_k})$ where $g_k = g^{y_k}$
- $\sigma \leftarrow (h, \sigma_2 / \prod_{k\in[\ell]} g_k^{r_k}) = (h, h^{x + \sum_{k\in[\ell]} m_k y_k})$

– $\mathsf{RS.Rerand}(\sigma, \Delta_r, \Delta_u) \rightarrow \sigma'$: Parse σ as (σ_1, σ_2) Set $\sigma_1' \leftarrow \sigma_1^{\Delta_u}$ Set $\sigma_2' \leftarrow (\sigma_2 \cdot \sigma_1^{\Delta_r})^{\Delta_u}$ Return $\sigma' \leftarrow (\sigma_1', \sigma_2')$

– $\mathsf{RS.Ver}(\mathsf{vk}, \mathsf{cm}, \sigma) \rightarrow \{0,1\}$: Parse σ as (σ_1, σ_2), The prover $\mathcal{P}$ runs a Proof of Knowledge protocol with the following relation

$$\mathcal{R} \leftarrow \mathsf{PoK}\{(m_1, \ldots, m_\ell, r + \Delta_r) :$$

$$e(\sigma_2', \tilde{g}) = e(\sigma_1', \mathsf{vk}) \cdot e(\sigma_1', \widetilde{\mathsf{cm}}') \quad \wedge \quad e(\mathsf{cm}', \tilde{g}) = e(g, \widetilde{\mathsf{cm}}') \quad \wedge \quad \mathsf{cm}' = g^{r+\Delta_r} \prod_{i=1}^{\ell} g_i^{m_i}\}$$

6.3 T-SIRIS: Threshold Sybil-Resistant Identity System

Building on our threshold PS signature scheme, we now present T-SIRIS, a complete threshold sybil-resistant identity system. T-SIRIS integrates threshold issuance, similar to Coconut [SABB+20], with credential relationship binding nullifiers (Chapter 5) to prevent sybil attacks (formalized in Definition [Sybil Resistance]), distinguishing it as a robust SSI solution. T-SIRIS integrates four key components:

- **Threshold Key Generation**: Uses distributed key generation (DKG) with Shamir secret sharing for decentralized trust across n issuers.
- **Threshold Issuance**: Extends MIMC-ABC (Chapter 4) to issue credentials with t of n issuers, ensuring robustness.
- **Deduplication**: Employs nullifiers from Chapter 5 for efficient sybil resistance without costly operations.
- **Expressive Proofs**: Leverages Chapter 3's ABCs for privacy-preserving predicate verification (e.g., range proofs).

We assume up to $t-1$ malicious issuers in an n-issuer system, where t is the threshold. These malicious issuers may:

- Deviate from the protocol specification
- Collude to attempt to forge credentials
- Attempt to track or deanonymize users
- Refuse to participate in the issuance protocol

We also assume users may attempt sybil attacks by requesting multiple credentials for the same context. However, the adversary cannot break the underlying cryptographic assumptions (DL and q-DDHI).

A tSIRIS system consists of the following probabilistic polynomial-time (PPT) algorithms, parameterized by a security parameter λ and attribute vector length ℓ. We use $\mathsf{RS.Setup}, \mathsf{RS.Rand}, \mathsf{RS.Ver}$ from the rerandomizable signature scheme.

Definition 6.3 (tSIRIS).

- $\mathsf{RS.Setup}(1^\lambda) \to \mathsf{pp}$*: Outputs public parameters* pp*, including a bilinear group* $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, p)$.
- $\mathsf{tKeyGen}(\mathsf{pp}, \ell, t, n) \to (\mathsf{ck}, \mathsf{vk}, \{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i\in[n]})$*: Generates public commitment and verification keys* ck, vk *and distributes secret/verification key shares to* n *issuers using a* (t, n)*-threshold scheme.*
- $\mathsf{Dedup}(\mathsf{cred_m}, \mathsf{cm_c}, \mathsf{nullifier}, \pi_{\mathsf{nullifier}}, \mathcal{T}) \to (\{0,1\}, T')$: *Takes input the user master credential* $(\mathsf{cred_m} = \sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{usk_m})$ *and context commitment* $\mathsf{cm_c}$*, the nullifier* $\mathsf{nullifier}$ *and its proof of correctness* $\pi_{\mathsf{nullifier}}$ *and the redeemed credential list* $\mathcal{T}$*. Outputs 0 if the proof fails or outputs 1 and updates the credential list* T *with* $\mathsf{nullifier}$
- $(\mathsf{tObtainMaster}(\mathsf{attrs}, \mathsf{ck}, h, \{\mathsf{vk}_i\}_{i\in S}), \mathsf{tIssueMaster}(\{(\mathsf{sk}_i)\}_{i\in S}, \{\mathsf{cm}_k, \pi_k\}_{k\in[\ell]})) \to (\mathsf{cred_m}, \perp)$*: an interactive protocol between a user running* $\mathsf{tObtainMaster}$ *from a subset* S *of issuers where* $|S| \geq t$ *with each issuer running* $\mathsf{tIssueMaster}$*.* $\mathsf{tObtainMaster}$ *takes in the user's attributes* attrs*, commitment key* ck *and verification key shares for at least* S *nodes* $\{\mathsf{vk}_i\}_{i\in S}$*.* $\mathsf{tIssueMaster}$ *is run at least* $|S|$ *times, takes input the signers shared secret key* sk_i*, commitment and proof pair for each message* $\{\mathsf{cm}_k, \pi_k\}_{k\in[\ell]}$*. Outputs* $\mathsf{cred_m}$ *to the user and* $\perp$ *to itself.*
- $(\mathsf{tObtainContext}(\mathsf{cred_m}, \mathsf{attrs}, \mathsf{ck}, \{\mathsf{vk}_i\}_{i\in S}), \mathsf{tIssueContext}(\{(\mathsf{sk}_i)\}_{i\in S}, \{\mathsf{cm}_k, \pi_k\}_{k\in[\ell]}, \mathsf{aux})) \to (\mathsf{cred_c}, \perp)$*: an interactive protocol between a user running* $\mathsf{tObtainContext}$ *from a subset* S *of issuers where*

$|S| \geq t$ *with each issuer running* tIssueContext. tObtainContext *takes in the user's master credential* $\mathsf{cred_m}$*, new attributes* attrs*, commitment key* ck *and verification key shares for at least* S *nodes* $\{\mathsf{vk}_i\}_{i\in S}$*.* tIssueContext *is run at least* $|S|$ *times, takes input the signers shared secret key* sk_i*, commitment and proof pair for each message* $\{\mathsf{cm}_k, \pi_k\}_{k\in[\ell]}$ *and auxiliary information* aux *e.g. outputs* $\mathsf{nullifier}, \pi_{\mathsf{nullifier}}, \mathcal{T}$ *from* Dedup *and* $\mathsf{cred_m}, \pi_{\mathsf{verify}}$ *from* $\mathsf{RS.Ver}(\mathsf{cred_m}, \mathsf{vk})$ *for* $\mathsf{cred_m}$*. Outputs* $\mathsf{cred_c}$ *to the user and* $\perp$ *to itself.*

– $(\mathsf{tShow}(\mathsf{cred}, \mathsf{r}, \phi), \mathsf{Verify}(\mathsf{cred}', \pi)) \rightarrow \{0, 1\}$*: An interactive protocol between a user and verifier. The user inputs a credential* $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{r})$ *and predicate for verification* ϕ*.* RS.Verify *takes input the rerandomized credential* cred' *and proof* π *it satisfies* ϕ*. The verifier outputs 1 if valid, 0 otherwise.*

Definition 6.4 (Threshold Unforgeability). *Threshold Unforgeability captures that with* $t-1$ *corrupt issuers, an adversary cannot forge valid credentials. We define the following experiment:*

Experiment $\mathsf{Exp}^{\mathsf{unf}}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda)$:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ck}, \mathsf{vk}, \{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i\in[n]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \ell, t, n)$
3. *Let* C *be a set of corrupted issuers where* $|C| \leq t-1$
4. $\mathcal{A}$ *is given* $\{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i\in C}, \{\mathsf{vk}_i\}_{i\in[n]\setminus C}$
5. $\mathcal{A}$ *has access to credential issuance oracles* $\mathcal{O}_{obtain},\ \mathcal{O}_{show}$
6. $\mathcal{A}$ *outputs* $(\mathsf{cred}', \phi, \pi)$
7. *The experiment returns 1 if:*
   – $\mathsf{Verify}(\mathsf{cred}', \phi, \pi) = 1$
   – cred' *was not legitimately issued to a corrupted user*
   – $\phi(\vec{m}) = 0$*, where* $\vec{m}$ *are the attributes in* cred'

A T-SIRIS scheme satisfies threshold unforgeability if for any PPT adversary $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{Exp}^{\mathsf{unf}}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

6.3.1 Sybil Resistance

We formalize sybil resistance, capturing that no user can obtain multiple credentials for the same context, even when interacting with multiple (potentially corrupted) subsets of issuers:

Definition 6.5 (Sybil Resistance). *We formalize sybil resistance as a user unable to obtain multiple credentials for the same context even when interacting with multiple (potentially corrupt) subset of issuers:*

Experiment $\mathsf{Exp}^{\mathsf{sybil}}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda)$:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ck}, \mathsf{vk}, \{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i\in[n]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \ell, t, n)$
3. *Let* C *be a set of corrupted issuers where* $|C| \leq t-1$

4. *$\mathcal{A}$ is given* $\{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i \in C}, \{\mathsf{vk}_i\}_{i \in [n] \setminus C}$
5. *Initialize deduplication table* $\mathcal{T} = \emptyset$
6. *$\mathcal{A}$ interacts with credential issuance oracles*
7. *$\mathcal{A}$ outputs master credential* cred_{master}*, context* ctx *and two context credentials* $(\mathsf{cred}_1, \mathsf{cred}_2)$
8. *The experiment returns 1 if:*
   - $\mathsf{Verify}(\mathsf{cred}_{master}, \phi_{master}, \pi_{master}) = 1$
   - $\mathsf{Verify}(\mathsf{cred}_1, \phi_{\mathsf{ctx}}, \pi_1) = \mathsf{Verify}(\mathsf{cred}_2, \phi_{\mathsf{ctx}}, \pi_2) = 1$
   - cred_1 *and* cred_2 *have the same context* ctx
   - cred_1 *and* cred_2 *were issued using the same* cred_{master}
   - $\mathsf{cred}_1 \neq \mathsf{cred}_2$ *(i.e., they are distinct credentials)*

A T-SIRIS scheme satisfies sybil resistance if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that:*

$$\Pr[\mathsf{Exp}^{\mathsf{sybil}}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

6.3.2 Threshold Anonymity

We formalize threshold anonymity, capturing that credentials remain unlinkable and reveal nothing beyond what's explicitly proven, even when up to $t - 1$ issuers are corrupted:

Definition 6.6 (Threshold Anonymity). *We define the following experiment:*

Experiment $\mathsf{Exp}^{\mathsf{anon-b}}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda)$:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ck}, \mathsf{vk}, \{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i \in [n]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \ell, t, n)$
3. *Let C be a set of corrupted issuers where* $|C| \leq t - 1$
4. *$\mathcal{A}$ is given* $\{(\mathsf{sk}_i, \mathsf{vk}_i)\}_{i \in C}, \{\mathsf{vk}_i\}_{i \in [n] \setminus C}$
5. *For* $i \in \{0, 1\}$*:*
   - *User i obtains a master credential* cred^i_{master} *with attributes* attrs_i
   - *User i obtains a context credential* $\mathsf{cred}^i_{\mathsf{ctx}}$ *for context* ctx
6. *$\mathcal{A}$ outputs predicate ϕ such that* $\phi(\mathsf{attrs}_0) = \phi(\mathsf{attrs}_1) = 1$
7. $b \leftarrow\!\!\$\ \{0, 1\}$
8. *Generate* $(\mathsf{cred}', \pi) \leftarrow \mathsf{Show}(\mathsf{cred}^b_{\mathsf{ctx}}, \mathsf{usk}_b, \phi)$
9. $b' \leftarrow \mathcal{A}(\mathsf{cred}', \pi)$
10. *The experiment returns 1 if $b' = b$, otherwise 0*

A T-SIRIS scheme satisfies threshold anonymity if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl *such that:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{anon}-1}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda)=1]-\Pr[\mathsf{Exp}^{\mathsf{anon}-0}_{\mathcal{A},\mathsf{T\text{-}SIRIS}}(\lambda)=1]\right|\leq \mathsf{negl}(\lambda)$$

6.3.3 Notes on Security Definitions

These security definitions capture the unique challenges of the threshold setting:

1. **Threshold Unforgeability**: Ensures that even with $t-1$ corrupted issuers, an adversary cannot forge credentials or prove false statements about them.

2. **Sybil Resistance**: Guarantees that the nullifier-based deduplication mechanism works correctly even when interacting with different subsets of issuers, preventing users from obtaining multiple credentials for the same context.

3. **Threshold Anonymity**: Ensures that credential presentations reveal nothing beyond the proven statement, even when up to $t-1$ issuers collude. This is stronger than standard anonymity since it must account for partial issuer corruption.

The deduplication table $\mathcal{T}$ should be replicated across all issuers or maintained through a consensus protocol.

6.4 Our Construction

6.4.1 Overview

T-SIRIS operates in two credential issuance phases: master credential issuance and context credential issuance. The master credential establishes the user's base identity with a secret key s, while context credentials are bound to this master credential through nullifiers derived from s and a context ctx. The nullifier mechanism prevents sybil attacks by ensuring users can obtain only one credential per context.

Freshness The current instantiation uses an interactive three-move Σ-protocols and as such the challenge guarantees freshness. Our system can be non-interactive via the Fiat-Shamir transform. To prevent replay attacks, the verifiers must store a collection of the public statements and proof from previous verifications.

Credential The Credential cred is a rerandomizable threshold signature over a commitment. The randomness of the commitment acts as a private key, provided the user can verify their credential with their commitment and prove knowledge of the opening of the commitment, they must know the randomness e.g. secret key that created the credential.

Master Credential Issuance Since we are optimizing for a private, accountable system, we make a small trade-off in privacy during Master Credential generation to satisfy the efficiency and private accountability requirements of an organization or government deploying this system today. We note that we can swap our methods for complete privacy using credential oracles like [ZMM+20, CDH+25, BCJ+24, RWG+18] with a reduction in security and accountability. During Master Credential registration, we enforce the user's Master Credential to be issued in a semi-trusted process where the user verifies their identity to the registration authority during its issuance. This enables Sybil resistance on the user identifier, stronger security for their VRF key s discussed below 6.4.2, which is used to attach other credentials to it. The Registration Authority keeps track of user identities and can request their revocation from the Accountability Authority with techniques from [DGK+20]. At the end of registration, the user has a master credential $\mathsf{cred}_\mathsf{m} = (\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{vk}_\mathsf{m})$, including a rerandomizable, threshold-signature over commitment.

6.4.2 Multi-Party Protocol for Secure Nullifier Key Generation

The VRF key s is embedded in the master credential and used to generate nullifiers for every context credential. Therefore, s must be generated in a way that prevents the user from stealing someone else's s and embedding it in their master credential (and therefore generating their nullifiers and abusing the system), as well as ensuring the issuers can't modify or create it maliciously to attempt to break anonymity later. In the single issuer protocol, the user commits to their s_1, $\mathsf{cm}_1 = \mathsf{CM.Com}([s_1]; r)$[1] and shares with the issuer along with a proof of its opening. Issuer verifies the proof and generates s_2, commits to it without randomness, $\mathsf{cm}_2 = \mathsf{CM.Com}([s_2]; 0)$ and returns cm_2, s_2 to the user. The user aggregates $\mathsf{cm} = \mathsf{cm}_1 \cdot \mathsf{cm}_2$ and computes $s = s_1 + s_2$ and now has a fully formed $\mathsf{cm} = \mathsf{CM.Com}([s_1 + s_2]; r)$. We adjust slightly for the threshold scenario. The user computes s_1, $\mathsf{cm}_1 = \mathsf{CM.Com}([s_1]; r)$ and a proof of its opening. The issuers runs DKG with a subset $|S| \geq t$ of nodes to produce $\mathsf{vk} = g^{s_2}$ and each node computes shared $\{k_{2_i}\}_{i \in S}$ and shares with the user (For efficiency, the issuers can precompute these values in bulk and retain a pool to choose from s_2). The user computes s_2 by combining $|S| \geq t$ with $\mathsf{DKG.Combine}(\{\mathsf{s}_{2_i}\}_{\mathsf{i} \in \mathsf{S}})$, the user then computes $\mathsf{cm} = \mathsf{CM.Com}([s_1 + s_2]; r)$ and runs a sigma protocol to prove its correctness during credential generation.

$$\mathcal{R}_{\mathsf{DKG}} : \left\{ (\mathsf{cm}_1, \mathsf{cm}_s, \mathsf{vk}), (s_1, s_2, s, r) \middle| \begin{array}{l} \mathsf{cm}_{s_1} = \mathsf{CM.Com}([s_1]; r) \wedge \\ \mathsf{vk}_{s_2} = g^{s_2} \wedge \\ \mathsf{cm}_s = \mathsf{CM.Com}([s_1 + s_2]; r) \end{array} \right\}$$

The issuer learns the user's identity during registration but never learns the complete VRF key - the user maintains anonymity against a malicious issuer and the authority maintains unforgeability against a malicious user.

Context Credential Issuance A context credential is linked to a master credential with the same committed identifier e.g., $\mathsf{cm}_\mathsf{m} = \mathsf{CM.Com}([id, \ldots]; r_1), \mathsf{cm}_\mathsf{c} = \mathsf{CM.Com}([id, \ldots]; r_2)$ kept secret during the issuance process. A context credential issuer may be the same threshold committee, but may also be in the form of a credential oracle [ZMM+20, CDH+25, BCJ+24, RWG+18]. The context credential issuance criteria will vary depending on their individual requirements; we represent this as the predicate ϕ that should be satisfied by the user during issuance. For example, perhaps a user successfully completed their driving test and should be issued a driving license. A user interacting with the DMV should first verify their master credential (e.g. they have a valid passport), The user also commits to their identifier $\mathsf{cm}_\mathsf{c} = \mathsf{CM.Com}([id, \mathsf{ctx}_\mathsf{c}, \ldots]; r_c)$ and proves their id is consistent between the two commitments. The user proves this relation:

$$\mathcal{R}_\phi : \left\{ \begin{array}{l} (\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{cm}_\mathsf{c}), \\ (id, s, \mathsf{ctx}_\mathsf{m}, \mathsf{ctx}_\mathsf{c}, r_m, r_c) \end{array} \middle| \begin{array}{l} \mathsf{RS.Ver}(\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{vk}_\mathsf{m}) = 1 \wedge \\ \mathsf{cm}_\mathsf{m} = \mathsf{CM.Com}([id, \mathsf{ctx}_\mathsf{m}, s, \ldots]; r_m) \ \wedge \mathsf{ctx}_\mathsf{m} = \texttt{"master"} \wedge \\ \mathsf{cm}_\mathsf{c} = \mathsf{CM.Com}([id, \mathsf{ctx}_\mathsf{c}, \ldots]; r_c) \ \wedge \mathsf{ctx}_\mathsf{c} = \texttt{"DMV"} \end{array} \right\}$$

The Context Credential issuer verifies the correctness of Π_ϕ and runs their signing protocol over cm_c.

Sybil Resistance with our Nullifier

In the above scenario, to enforce sybil resistance of the user's driver license, the DMV authority could request to retain id in their system and check that value against another user with id requesting a credential, upholding accountability but breaking the user's anonymity and open forgability vectors, breaking system security. Rather, the user generates an anonymous deterministic (pseudorandom) nullifier with the scheme 5.5.4 as $\mathsf{nullifier} = g^{1/s+ctx} \leftarrow \mathsf{dVRF.Eval}(s, ctx)$. The nullifier's proof of correctness $\pi \leftarrow \mathsf{dVRF.Prove}(\mathsf{cm}_\mathsf{m}, \mathsf{cm}_\mathsf{c}, s, ctx, \mathsf{nullifier})$, outlined here: 5.5.3, ensures it was generated by s in the master credential and ctx_c from the Context Credential. To prevent replay attacks, $\mathsf{nullifier}$ is an input into the context credential issuance process π is combined with the previous Σ-protocol.

$$\mathcal{R}_\phi : \left\{ \begin{array}{l} (\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{cm}_\mathsf{c}, \mathsf{nullifier}), \\ (id, s, \mathsf{ctx}_\mathsf{m}, \mathsf{ctx}_\mathsf{c}, r_m, r_c) \end{array} \middle| \begin{array}{l} \mathsf{RS.Ver}(\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{vk}_\mathsf{m}) = 1 \wedge \\ \mathsf{cm}_\mathsf{m} = \mathsf{CM.Com}([id, \mathsf{ctx}_\mathsf{m}, s, \ldots]; r_m) \ \wedge \mathsf{ctx}_\mathsf{m} = \texttt{"master"} \wedge \\ \mathsf{cm}_\mathsf{c} = \mathsf{CM.Com}([id, \mathsf{ctx}_\mathsf{c}, \ldots]; r_c) \ \wedge \mathsf{ctx}_\mathsf{c} = \texttt{"DMV"} \wedge \\ \mathsf{nullifier} = g^{1/s+ctxc} \end{array} \right\}$$

[1] cm contains more than just the VRF key s, we simplify in this explanation for brevity

Revocation

A popular efficient accountable-privacy pattern for revocation is seen in anonymous payments [DGK+20] and private identity systems [**?**]: on registration, a user encrypts a secret key with a separate (preferably threshold) authority, e.g. Audit Authority and stores the encryption on the registration system within the original user profile. Should it require revocation, the Registration System provides the encryption and reason for revocation to the auditors, the auditors then decrypt it and add the key to a revocation list such as an accumulator that enables efficient zero-knowledge non-membership proofs [JML24, VB22] which cost 7.4ms for proof generation and 11.2ms for proof verification, adding some overhead but yet still practical. We can extend this pattern to enable revocation of individual context credentials - the auditor receives the request to revoke a specific context for an identity, the auditor decrypts s and computes $\mathsf{nullifier} = g^{1/(s+ctx)}$ and includes that in an accumulator.

6.4.3 Protocol Details

OrgKeyGen $\mathsf{OrgKeyGen}(1^\lambda, \ell, t, n) \to (\mathsf{BG}, \mathsf{ck}, \mathsf{vk}, \{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[\ell]}, \{[s_2]_i\}_m$ for $i \in |n|, m$ is arbitrarily large): The trusted setup runs the following algorithms and distributes the key shares to the threshold nodes.

- $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}, p) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda)$
- $(\mathsf{ck}, \mathsf{vk}, \{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[\ell]}) \leftarrow \mathsf{tDistKeyGen}(1^\lambda, t+1, n, \ell, \mathsf{BG})$
- Generate a pool of arbitrary size VRF key shares $\{\{s_2\}_i, \mathsf{vk}_{s_2}\}_m = \{\{s_2\}_i, \mathsf{vk}_{s_2}\}_1, \ldots, \{\{s_2\}_i, \mathsf{vk}_{s_2}\}_m$ for $i \in |n|$ and m is arbitrary size. The committee can replenish the pool periodically.
- Returns $(\mathsf{BG}, \mathsf{ck}, \mathsf{vk}, \{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[\ell]}, \{[s_2]_i\}_m$ for $i \in |n|)$

(ObtainMaster, IssueMaster) The user and issuers interact, user executes Obtain with their attributes to compute the outputs for issuing. Issue is run by the threshold nodes to issue the signature shares.

Obtain: $\mathsf{ObtainMaster}(\mathsf{attrs})$

1. User samples $s_1, r_s \leftarrow\!\$\ \mathbb{Z}_p$, $\mathsf{cm}_{s_1} = \mathsf{CM.Com}([s_1]; r_s)$
2. User interacts with the threshold to receive h and a subset of $[s_{2_i}]_{i\in S}$ shares $|S| \geq t$ and vk_{s_2}, user combines $s_2 \leftarrow \mathsf{DKG.Combine}([s_{2_i}]_{i\in S})$ and verifies $g^{s_2} = \mathsf{vk}_{s_2}$.
3. User runs $\mathsf{tPrepare}(\mathsf{ck}, h, \mathsf{attrs}) \to (\{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h)$, which includes the proof the user combined s correctly. $\mathsf{cm}_s = \mathsf{CM.Com}([s_1 + s_2]; r_s) = h^{s_1+s_2} g^{r_s}$ and proof of correctness $\Pi^{\mathcal{R}_{\mathsf{DKG}}}(\mathsf{cm}_{s_1}, \mathsf{vk}_{s_2}, \mathsf{cm}_s)$
4. inputs all cm_k for Issue

IssueMaster: $\mathsf{IssueMaster}(\{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h) \to \sigma$: A subset of $|S|$ issuers receive signing requests from the user.

1. $|S|$ issuers run $\mathsf{tShareSign}(\mathsf{ck}, \mathsf{sk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, h) \to [\sigma^*]_i$:
2. User receives $[\sigma^*]_i$ and runs $\mathsf{tShareVer}(\mathsf{ck}, \mathsf{vk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, [\sigma^*]_i; h) \to \{0, 1\}$: If output is 1 for $\geq t$ shares, User aggregates.
3. $\mathsf{tAggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, \{r_k\}_{k\in[\ell]}) \to \sigma$:

(ObtainContext, IssueContext) The user and issuers interact, user executes Obtain with their master credential and context attributes to compute the outputs for issuing. Issue is run by the threshold nodes to issue the signature shares.

ObtainContext: $\mathsf{ObtainContext}(\mathsf{attrs}, \mathsf{cred_m}, \phi)$

1. User samples $\Delta_r, \Delta_u \leftarrow\$ \mathbb{Z}_p$, rerandomizes $\mathsf{RS.Rand}(\sigma_\mathsf{m}, \mathsf{cm_m}) \rightarrow \sigma'_\mathsf{m}, \mathsf{cm_m}'$
2. User computes $\mathsf{nullifier}, \Pi^{\mathsf{R_{credc}}}, \mathsf{nullifier} \leftarrow \mathsf{dVRF.Eval}(s, \mathsf{ctx})$ and $\Pi^{\mathsf{R_{credc}}} \leftarrow \mathsf{dVRF.Prove}(\mathsf{cm_m}, \mathsf{cm_c}, s, \mathsf{ctx}, \mathsf{nullifier})$, during the same execution with the verifier, user computes $\mathsf{tPrepare}$ to output $(\{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h)$
3. inputs all cm_k for Issue

IssueContext: $\mathsf{Issue}(\mathsf{nullifier}, \Pi^{\mathsf{R_{credc}}}, \{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h) \rightarrow \sigma$: A subset of $|S|$ issuers receive signing requests from the user.

1. $|S|$ issuers run
   (a) $\mathsf{ZK.Verify}(\mathsf{cm_m}, \mathsf{cm_c}, \mathsf{nullifier}, \Pi^{\mathsf{R_{credc}}})$ and $\mathsf{ZK.Verify}(\{\mathsf{cm}_k, \Pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]})$[2]. The cm_k committing to ctx opens to the same index of $\mathsf{cm_c}$ which the issuers will verify.
   (b) $\mathsf{tShareSign}(\mathsf{ck}, \mathsf{sk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, h) \rightarrow [\sigma^*]_i$:
2. User receives $[\sigma^*]_i$ and runs $\mathsf{tShareVer}(\mathsf{ck}, \mathsf{vk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, [\sigma^*]_i; h) \rightarrow \{0, 1\}$: If output is 1 for $\geq t$ shares, User aggregates.
3. $\mathsf{tAggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, \{r_k\}_{k\in[\ell]}) \rightarrow \sigma$:

(Show, Verify) Show and Verify follow from the non-threshold ABC system 3.5.4. The user runs $\mathsf{RS.Rand}(\sigma, \mathsf{cm}) \rightarrow \sigma', \mathsf{cm}'$ then runs $\mathsf{RS.Ver}(\mathsf{vk}, \mathsf{cm}', \sigma')$. Before rerandomization, the commitment is without randomness $\mathsf{cm} = \mathsf{CM.Com}([\mathsf{attrs}]; 0)$ and the user rerandomizes it before verification.

6.4.4 Security Analysis

We analyze the security of T-SIRIS by demonstrating how it inherits and extends the security properties of its component schemes when deployed in a threshold setting with up to $t-1$ malicious issuers.

Unforgeability

Theorem 6.7 (Unforgeability). *If the threshold signature scheme is EUF-CMA secure and the MIMC-ABC system is unforgeable, then* T-SIRIS *is unforgeable against up to $t-1$ malicious issuers.*

Proof (Sketch). The unforgeability of T-SIRIS reduces to the security of two underlying components:

1. **Threshold signature unforgeability:** The threshold PS signature construction in Section 6.2.1 inherits the EUF-CMA security of the standard PS signature scheme from Chapter 3. With at most $t-1$ corrupted issuers, an adversary cannot forge a valid signature without the participation of at least one honest issuer.
2. **Credential verification unforgeability:** The verification of credential predicates reduces to the unforgeability of the MIMC-ABC system (Chapter 4), which prevents adversaries from proving false statements about credential attributes.

Since both underlying components are secure under their respective assumptions, a successful attack against T-SIRIS would require breaking at least one of these primitives, which occurs with at most negligible probability.

[2] Both proofs use the same verifier challenge, they are separated here to aid understanding

Sybil Resistance

Theorem 6.8 (Sybil Resistance). *If the deterministic nullifier scheme from Chapter 5 satisfies uniqueness, then* T-SIRIS *is sybil-resistant.*

Proof (Sketch). The sybil resistance of T-SIRIS directly reduces to the uniqueness property of the deterministic Credential Relationship Binding Nullifier (CRBN) scheme presented in Section 5.5.4.

For each user-context pair (s, ctx), where s is the master credential key and ctx is the context identifier, the deterministic nullifier $y = g^{1/(s+\mathsf{ctx})}$ has proven uniqueness under the q-DDHI assumption. The deduplication mechanism ensures this nullifier is used only once per context.

Even in a threshold setting where some issuers are corrupted, as long as the deduplication table is consistently maintained across honest issuers, no user can obtain multiple credentials for the same context without breaking the cryptographic assumptions of the nullifier scheme.

Threshold Anonymity

Theorem 6.9 (Threshold Anonymity). T-SIRIS *provides threshold anonymity if the MIMC-ABC system is anonymous and the CRBN scheme preserves anonymity, even when up to* $t-1$ *issuers are corrupted.*

Proof (Sketch). Threshold anonymity combines two aspects:

1. **Credential anonymity:** From the MIMC-ABC system (Chapter 4), credential presentations reveal nothing about the user's identity beyond what is explicitly proven. The rerandomization of signatures ensures presentations are unlinkable.
2. **Nullifier anonymity:** The CRBN scheme (Chapter 5) ensures that nullifiers reveal nothing about the user's identity or master credential. For revocation purposes, the committed nullifier variant provides additional unlinkability.

In the threshold setting, these properties are preserved because:

- No subset of fewer than t issuers can reconstruct the master secret key
- The zero-knowledge proofs reveal nothing beyond statement validity
- The credential presentations use fresh randomness each time

Through a standard hybrid argument, we can show that the adversary's advantage in distinguishing between different users' credential presentations is negligible, even with up to $t-1$ corrupted issuers.

Unlinkability

Theorem 6.10 (Unlinkability). T-SIRIS *provides unlinkability if the CRBN scheme satisfies the unlinkability property and the rerandomizable signature scheme satisfies rerandomization indistinguishability.*

Proof (Sketch). Unlinkability in T-SIRIS ensures that multiple presentations of credentials from the same user cannot be linked, even across different verifiers or sessions. This property builds on:

1. **Signature rerandomization:** Each credential presentation uses fresh randomness, making signatures statistically independent and computationally indistinguishable from newly issued signatures.

2. **Committed nullifiers:** For ongoing presentations, we use the committed nullifier variant $\mathsf{cm}_y = g_3^{1/(s+\mathsf{ctx})} g^r$ with fresh randomness r, ensuring unlinkability while maintaining verifiability.

The threshold setting strengthens this property by distributing trust: even if $t-1$ issuers collude, they cannot link credential presentations without breaking the underlying cryptographic assumptions of the rerandomizable signature scheme or the commitment scheme.

6.5 Performance Evaluation

We evaluate the performance of T-SIRIS and its underlying cryptography by first benchmarking individual cryptographic operations from our threshold signature scheme, comparing it to the Threshold signature scheme from TACT [RAR+24]. We then evaluate the performance of T-SIRIS by comparing it to S3ID, the Threshold Identity system constructed from TACT.

6.5.1 Threshold Signature Evaluation

We first analyze the performance of the core cryptographic primitives: our threshold signature scheme versus the threshold anonymous counting token (TACT) scheme used in S3ID [RAR+24], a state-of-the-art Sybil Resistant, Threshold Anonymous Credential system with orthogonal techniques but different cryptography.

Scaling by attribute count (n) We evaluate performance with fixed threshold parameters ($N = 16, t = 9$) while varying attribute count n from 4 to 128, as shown in Figure 15

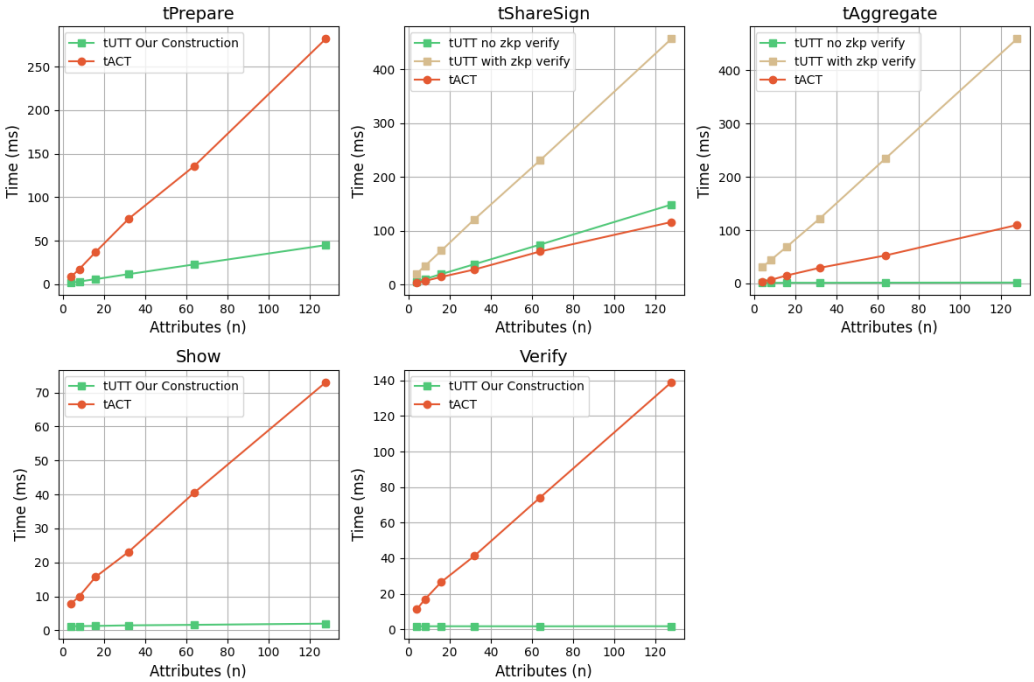

Fig. 15. Performance comparison when scaling by attribute count (n) with fixed $N = 16, t = 9$

– **tPrepare (Commitment Generation):** Our construction significantly outperforms tACT, achieving 6.0× faster operation at $n = 64$ (22.71ms vs. 135.91ms). This advantage stems from our optimized multi-scalar multiplication (MSM) implementation in Schnorr proofs, allowing sublinear scaling with increasing attribute count.

– **tShareSign (Signature Share Generation):** This is the only operation where tACT outperforms our construction (61.34ms vs. 244.22ms at $n = 64$), primarily because tACT doesn't

perform zero-knowledge verification of its messages during signing, both for the issuer verifying from the user, and the user verifying the individual signature shares. For completeness, we present both with and without ZKP verification variants of our algorithm.

– **tAggregate (Signature Aggregation):** Our construction achieves efficiency with 36.0× speedup at $n = 64$ (1.46ms vs. 52.55ms), we again leverage multi-scalar multiplication.

– **Show and Verify:** These operations demonstrate our advantage with nearly constant-time performance regardless of attribute count. At $n = 64$, our construction achieves 25.2× speedup for Show (1.61ms vs. 40.56ms) and 44.1× speedup for Verify (1.68ms vs. 74.07ms). This efficiency stems from our Schnorr ZKP approach, which avoids costly pairing checks per attribute or computing over $\mathbb{G}_T$ points.

The results in Figure 15 clearly illustrate how our construction maintains near-constant verification times regardless of attribute count, while tACT's performance degrades linearly with increasing attributes.

Scaling with Threshold Size (N) We now examine performance with fixed attribute count ($n = 16$) while varying threshold parameters across $N = 4$ ($t = 3$), $N = 16$ ($t = 9$), and $N = 64$ ($t = 33$). Key observations:

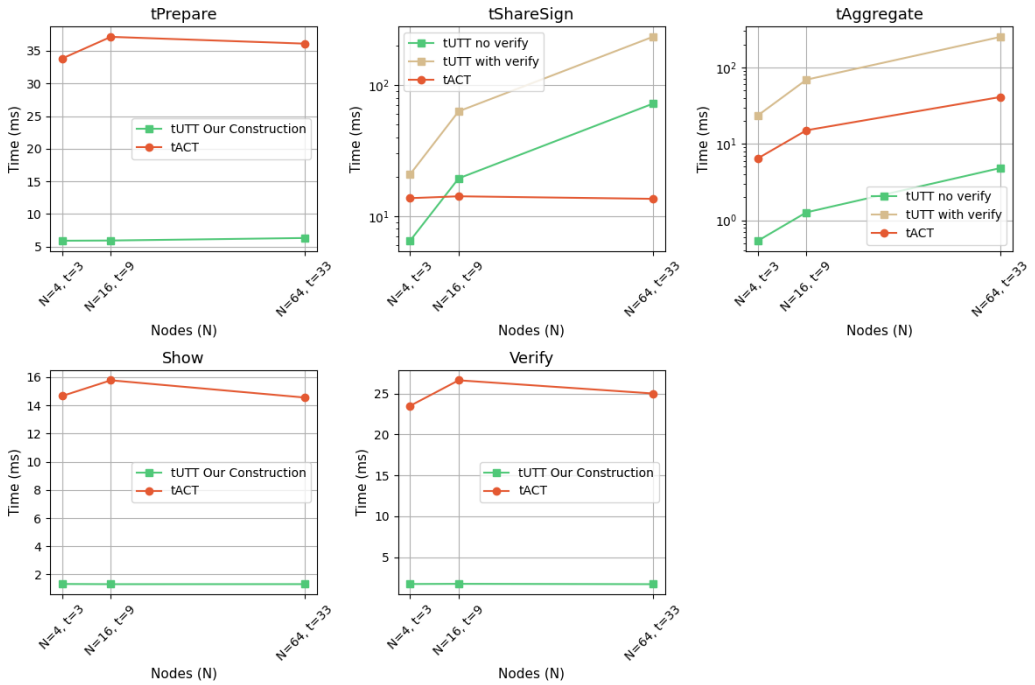

Fig. 16. Performance comparison when scaling by threshold size (N) with fixed $n = 16$

– **tShareSign Scalability:** Our construction's signing time increases with threshold size (21.19ms for $N = 4$ to 283.72ms for $N = 64$) due to our threshold key generation and signing approach. This contrasts with tACT's relatively stable performance (∼13-14ms regardless of N).

– **Constant-Time Operations:** For user-centric operations (tPrepare, Show, Verify), our construction maintains excellent performance regardless of threshold size. Verify times remain approximately 1.7ms across all N values, compared to tACT's ∼25ms.

– **Practical Implications:** These results suggest our construction is particularly well-suited for systems where credentials are frequently shown but rarely issued, as the verification performance advantage (up to 14.8× at $N = 64$) directly impacts end-user experience.

Operation	n=4	n=16	n=64
tPrepare (Ours)	1.65	6.16	22.71
Token Request	8.55	37.15	135.91
tShareSign (Ours)†	20.95	63.40	244.22
Issue	3.09	14.14	61.34
tAggregate (Ours)	0.99	1.10	1.46
(Aggr., Unblind)	3.92	15.04	52.55
Show (Ours)	1.26	1.35	1.61
Prove	7.90	15.78	40.56
Verify (Ours)	1.71	1.82	1.68
Verify	11.20	26.64	74.07

Fig. 17. tUTT Performance scaling with attribute count (n) for fixed $N = 16, t = 9$ (ms)

Operation	N=4	N=16	N=64
tPrepare (Ours)	5.97	6.16	5.88
Token Request	33.84	37.15	36.11
tShareSign (Ours)†	21.19	63.40	283.72
Issue	13.68	14.14	13.52
tAggregate (Ours)	0.51	1.10	4.89
(Aggr., Unblind)	6.46	15.04	41.02
Show (Ours)	1.33	1.35	1.33
Prove	14.67	15.78	14.55
Verify (Ours)	1.69	1.82	1.69
Verify	23.51	26.64	25.03

Fig. 18. tUTT Performance scaling with threshold size (N) for fixed $n = 16$ (ms)

Figure 16 highlights the key trade-off in our design: while tShareSign performance scales less favorably with increasing threshold size, our user-centric operations (particularly Show and Verify) maintain consistently superior performance regardless of committee size.

Performance Analysis Summary Our UTT (Ours) construction demonstrates superior performance in 4 out of 5 key operations, with notable advantages in:

1. **Verification efficiency:** Near-constant verification time regardless of attribute count, critical for real-world deployment scenarios.

2. **Show operation:** Consistently fast credential presentation (1.3-1.6ms), enabling responsive user experiences.

3. **Attribute scalability:** Excellent performance with large attribute sets, allowing for richer credential schemas without performance penalties.

The one trade-off is in signature share generation (tShareSign), where our construction prioritizes security through comprehensive zero-knowledge proofs at the cost of higher computational overhead. However, this operation typically occurs less frequently in practical deployments compared to verification operations.

6.5.2 Threshold Identity System Evaluation

We now analyze the complete T-SIRIS system's performance. Figure 19 illustrates how each system operation scales with attribute count across different threshold configurations.

The results reveal several important characteristics of our system:

– **User-side operations** (Obtain Master, Obtain Context, Show) scale linearly with attribute count, with Obtain Master at approximately $60ms$ for 128 attributes, enabling efficient user-side operation.

– **Issuer operations** (Issue Master, Issue Context) show notably different scaling behavior across threshold configurations. With 64 attributes, Issue Master takes approximately 200ms at N=4, 800ms at N=16, and 3000ms at N=64, highlighting the trade-off between decentralization and issuer-side performance. It's important to note that we do not issue with parallel computation and thus issuance time could be greatly reduced. We are opting for the worst-case scenario.

– **Verification operation** demonstrates near-constant time performance (approximately 1.7ms) regardless of attribute count or threshold configuration, crucial for high-throughput verification scenarios.

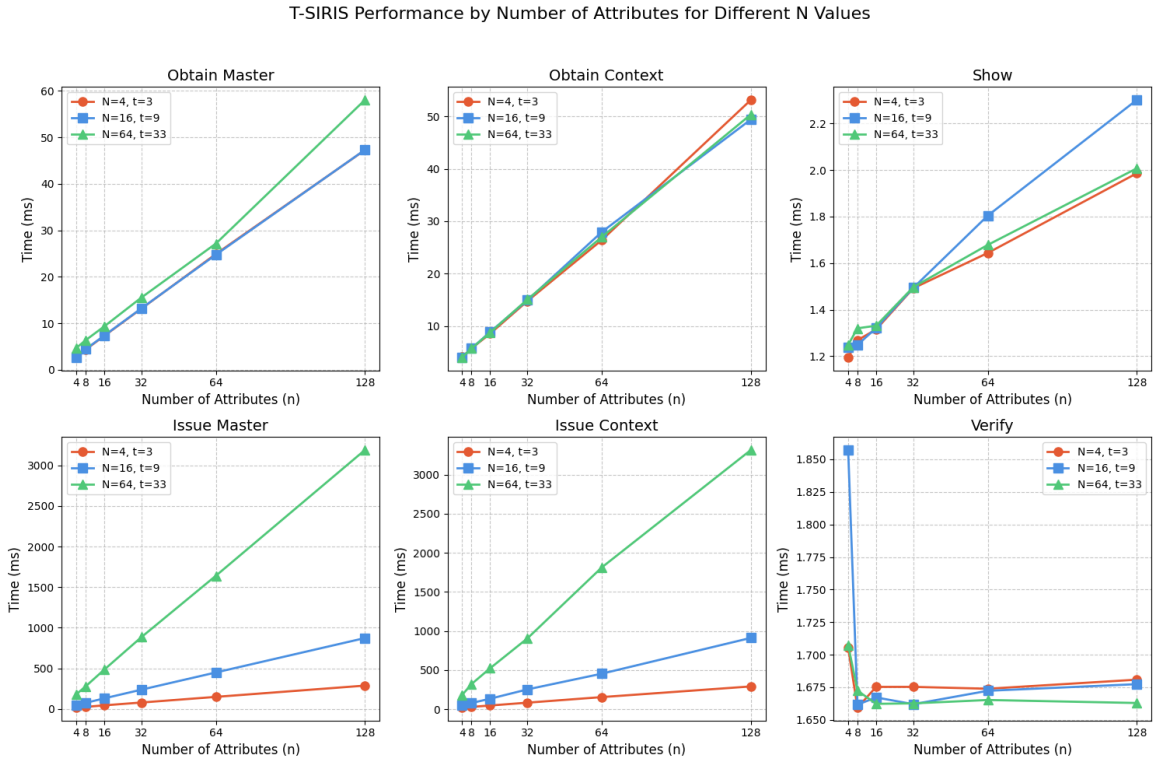

Fig. 19. T-SIRIS performance scaling with attribute count across operations

These measurements demonstrate that T-SIRIS achieves efficient credential presentation while maintaining acceptable issuance costs even with large threshold committees, making it practical for real-world deployment.

6.5.3 Threshold Identity System Performance Comparison with S3ID

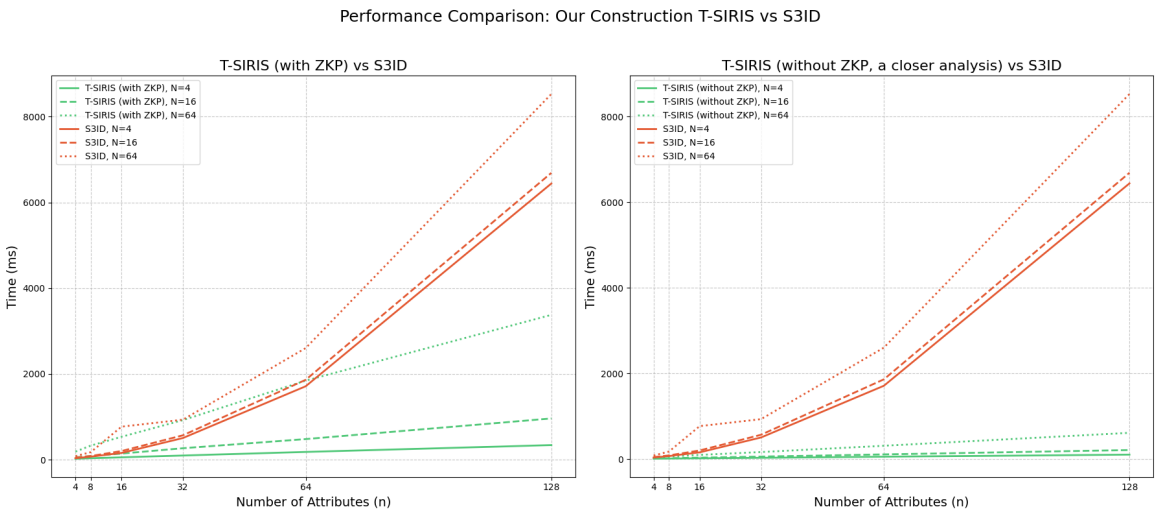

Fig. 20. End-to-end performance comparison between T-SIRIS and S3ID

We present a performance comparison between T-SIRIS and S3ID. S3ID's algorithm Dedup combines the credential request and issuance process, with sybil resistance and verification. We combined our algorithm benchmarks to compare against this and plotted this figure: 20.

The comparison reveals several significant advantages of our approach:

- **Verification efficiency**: T-SIRIS maintains consistent sub-2ms verification time regardless of attribute count, while S3ID's verification time increases linearly, reaching approximately 25ms for just 16 attributes. This 12.5x improvement in verification throughput is critical for large-scale deployments.
- **Efficient Sybil resistance**: Our CRBN-based nullifier mechanism adds only 2.49ms overhead during credential issuance, approximately 5x faster than S3ID's token-based deduplication approach (12.8ms), while providing equivalent Sybil-resistance guarantees.
- **Scalable attribute support**: While both systems show linear scaling with attribute count, T-SIRIS demonstrates significantly better slopes across all operations, with the greatest advantage in verification operations (44.1x at 64 attributes).

Even when issuer-side ZKP verification is included (our full security model), T-SIRIS outperforms S3ID in end-to-end credential issuance and verification scenarios. The performance gap widens further at higher attribute counts, making T-SIRIS suitable for complex credential schemas with many attributes and verifying multiple credentials together.

Chapter 7

Open Source Anonymous Credentials Benchmark Library

7.0.4 Core Research Problem

The central challenge we're solving is: How can we establish a fair, standardized, and practical performance evaluation framework for Attribute-Based Anonymous Credential (ABC) schemes to accelerate their real-world adoption?

There are two major gaps in the current literature:

1. Efficiency Comparisons: What are the real performance differences between older and newer anonymous credential schemes when tested under identical conditions (e.g., same hardware, security parameters, and libraries)?

2. Impact of Optimizations: How do cryptographic library optimizations—like Multi-Scalar Multiplication (MSM), batch verification, or Miller-Loop pairing optimizations—affect the practical efficiency of these schemes compared to their theoretical complexity?

This problem arises because current evaluations use different programming languages (Python, C++, Rust), libraries (Arkworks, Ethereum PyEcc), and optimization choices, creating incomparable results. Consistent benchmarks are needed to compare and select a scheme fairly.

7.0.5 Core Contributions

1. Rich Experimental Data for the most popular state-of-the-art ABCs and an Open-source Anonymous Credentials Benchmark Library written in Rust for ongoing fair comparisons.

2. Practical Efficiency Optimization Data: Showing the impact of Schnorr proofs using MSM scaling sublinearly rather than linearly as previously thought.

3. Random Linear Combination Proof Aggregation (Schnorr proofs, pairings)

4. Use-case experimental data for zero-knowledge proof statements

   Speedup techniques in rust library

Anonymous Credentials serve as the backbone of privacy-preserving identity systems.

Since their inception, Anonymous Credentials have primarily been academic works; however, with the increasing need for governments and organizations to migrate secure digital identity infrastructure, for example, in the US, 13 states have interoperable mobile digital driving licenses [AAM] with 30 more states in consideration, in addition to the EU's eIDAS framework requiring all member

states to have such infrastructure by 2026 [Eur24]. In light of this, there are products from Microsoft [Mic], AWS [AWS], and [Doc] enabling this technology in addition to standardisation efforts at W3C [WGD+25, WMD+22]

Therefore, the urgent question is, what is this technology's privacy/security cost? There are constant trade-offs between privacy, security, functionality, and efficiency. Improving privacy and security comes at the expense of functionality and efficiency, and vice versa. To answer this, we have built an Open-Source Anonymous Credentials Benchmark Library and (thus far) built 6 different schemes with the goal of fair comparison. We show why previous benchmark analyses are not fair and furthermore we show our analysis of using efficient practical cryptography libraries.

Previous deployments like IBM's Idemix [CVH02] and IRMA [AJ13] based on CL-Signatures [CVH02,CL03], Hyperledger Fabric [ABB+18] based on BBS+ [ASM06], Microsoft's U-Prove [CKL+16] based on Brands' signatures [Bra00], were deployed in pilot programs [CKL+16], and proved that these systems were secure, private and feature-rich, but inefficient for widespread adoption. Their implementations were based on early constructions we label as "previous generation" and lacked efficient credential verification procedures required for the expected user experience needed for large-scale uptake, which we illustrate below.

IRMA, Idemix and U-Prove: IRMA [AJ13] presents the Show+Verify for credential with 3 attributes costs about 1.3seconds (Intel Core i7-8650U CPU with 16 GB of RAM GNU/Linux) in [ZKS+21]

[VS16]: presents a more thorough evaluation of the Show+Verify algorithms: Idemix presents Show + Verify algorithm for 1,2,3 credentials as 110ms, 220ms, 450ms U-Prove presents Show + Verify algorithm for 1,2,3 credentials as 180ms, 460ms, 600ms Furthermore, they show functionality use-case like selective disclosure (400ms for 1 attribute selective disclosure and 670ms for 2 attributes selective disclosure) and committed attribute equality (120ms).

The gap we see here is that - doesn't take batch verification into account. - selective disclosure costs minimal overhead. (under 1ms) - attribute equality costing minimal overhead (under 1ms) - Newer schemes [CDL16, TBA+22] reduce pairing operations in the signature construction itself, coupled with cryptography library improvements reduce all metrics significantly. This hasn't been illustrated properly.

There is a gap in practical evaluation analysis for newer state-of-the-art constructions which we provide benchmarks for.

For Anonymous Credential and Privacy-Preserving Identity there is no library to help people learn the differences between constructions, what they can use or can't use and how they differ from a practical / functional implementation standpoint.

Our contributions 1. built opensource library framework for SOTA anonymous credentials and benchmarked against each other in rust 2. Demonstrate that schnorr proofs which are usually "slandered" in academic papers for being linear in the size of the proven attributed are in-fact sublinear in practice due to MSM techniques 3. we show the speedup from batch / windowing schnorr proofs 4. Show the computation reduction in pairings when using miller loop intermediate computation rather than computing and epxonentiating GT points

The Anonymous Credential field lacks standardized fair comparisons between the leading schemes (particularly BBS+ [ASM06] and PS [PS16]) and their newer variants [CDL16], [TBA+22]. Previous evaluations attempt to compare different schemes - although they might use consistent security parameters, variation in programming languages of the implementations (Python, C++, Rust), their underlying cryptography libraries (Arkworks, Ethereum PyEcc), and use of optimizations such as like Multi-Scalar Multiplication, Miller Loop Pairings, and Batch Techniques make fair comparison difficult to quantify.

Our first contribution is to show the key cryptographic operations for ABC systems and present the most efficient scheme in terms of the most used operation (Show + Verify). Our second

contribution is demonstrating the gap between theoretical and practical cryptography by implementing speedup techniques for critical operations and demonstrating our practical analysis. Our analysis is useful outside of Anonymous Credentials.

1. First, we show that implementing a Schnorr protocol to prove knowledge of committed exponents scales sublinearly in the size of the attribute length (contrary to the theoretical analyses in most literature) - importantly, before making claims on performance enhancements against previous "linear" schemes, it's imperative for new schemes to compare under these assumptions rather than pure theoretic.

2. Second, we show a batch verification technique for multiple individual schnorr proofs improving verification time by 50%. Demonstrate a batch verification method of individual schnorr proofs (useful in the threshold system where a user commits to each message individually, the verifier can create random linear combination and verify that $(ps_u tt_t s_s rc_s igner_b atch_v erify)$

3. third, we show that pairing-based credentials significantly improve with optimized Miller-Loop implementations. Transforming a pairing equality check from this to this means a speedup of X.

7.1 Bridging the Gap with an Open-Source Library

Anonymous credential systems promise privacy-preserving identity solutions, yet practitioners often lack accessible tools to understand the practical differences between constructions like Schnorr-based proofs and pairing-based systems. Existing literature focuses heavily on theoretical complexity, leaving an educational gap in functional implementation details—such as what systems can or cannot do in real-world scenarios and how they perform under specific use cases. To address this, we developed an open-source library framework in Rust, implementing state-of-the-art (SOTA) anonymous credential constructions and benchmarking them against each other.

This library serves as a standardized toolkit, enabling practitioners to explore how different systems handle use cases like decentralized identity or Sybil-resistant voting. By providing empirical performance data alongside functional insights, it clarifies trade-offs that theoretical analyses often overlook. For example, while academic papers may emphasize asymptotic complexity, our benchmarks reveal how optimizations shift these boundaries in practice, making the library a valuable educational and practical resource.

Open-Source Benchmarking Framework

We developed the first standardized open-source tooling for fair comparison of anonymous credential schemes, tackling inconsistencies in previous academic evaluations. This framework empowers researchers and industry practitioners to make informed implementation decisions using quantitative performance metrics, moving beyond unfair practical claims or purely theoretical analysis. By enabling reliable and consistent assessments, it accelerates both research advancements and practical adoption of privacy-preserving technologies.

7.2 New Generation Anonymous Credentials

The performance comparison in Table 13 demonstrates that newer anonymous credential schemes, such as BBS+2016 [CDL16] and PS-UTT22 [TBA+22], outperform older schemes like BBS+06 [ASM06] and PS16 [PS16] by shifting zero-knowledge proofs from the target group ($\mathbb{G}_T$) to the source group ($\mathbb{G}_1$)[3]. For 30 attributes, older schemes require computing over 3ℓ, and 2ℓ $\mathbb{G}_T$ points respectively with Show and Verify times of 39.21 ms (BBS+06) and 32.78 ms (PS16). In comparison, newer schemes compute only in $\mathbb{G}_1$ and minimize pairings, achieving times of 5.72 ms and 5.38 ms—speedups of 6.85x and 6.09x, respectively. These improvements maintain full functionality, including selective disclosure and unlinkability, and rely on standard security assumptions, enhancing the practicality of anonymous credentials for digital identity systems.

[3] CL04 [CL04] is included in previous generation

Table 13. Performance Comparison of Anonymous Credential Schemes for Show and Verify Algorithms ($\ell = 30$ Attributes)

Scheme	ZKPoK	Show (ms)	Verify (ms)	Show + Verify (ms)	Speedup
BBS+06	$\mathbb{G}_T$	12.91	26.30	39.21	–
BBS+2016	$\mathbb{G}_1$	3.15	2.57	5.72	6.85x over BBS+ 06
PS16	$\mathbb{G}_T$	16.23	16.55	32.78	–
PS-UTT22	$\mathbb{G}_1$	1.59	3.79	5.38	6.09x over PS16

Threshold Do a Table comparing Threshold Keys from TACT, Us, and BBS+

Crypto Library Optimizations Do a Table showing Single Pairing Miller Loop + Final Exponentiation G1 Exponentiation, Addition G2 Exponentiation, Addition GT Exponentiation, Addition

7.2.1 ABC Performance Comparison

7.3 Practical Proof Analysis

7.3.1 Sigma Protocols

Many schemes refer to sigma protocol as having linear size proofs. While this is true in theory, using multi-scalar-multiplication, a popular algorithm in many cryptographic libraries, we show that sigma protocols are, in fact, sublinear rather than linear when message size doubles.

These findings support the hypothesis that practical efficiency is substantially better than theoretical complexity would suggest when using MSM in Schnorr protocols and thus the proof protocols in PS and BBS+ based anonymous credentials are sublinear in practice.

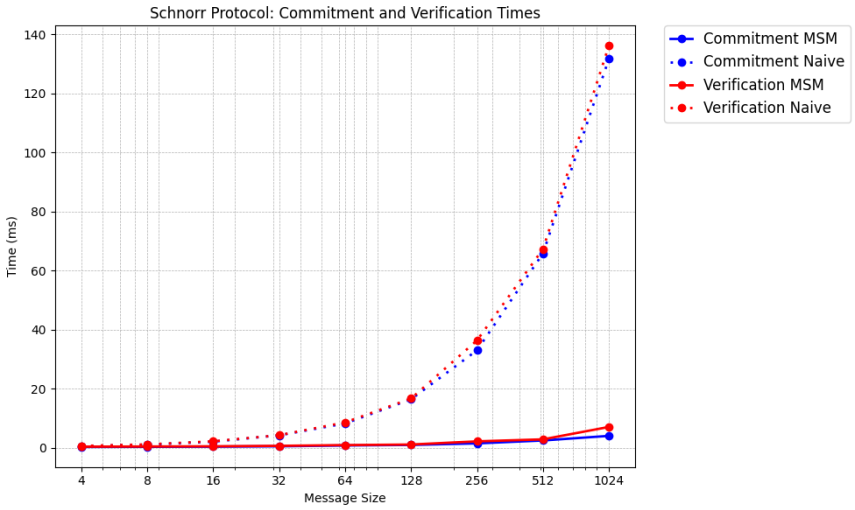

Fig. 21. Schnorr Protocol - Practical Benchmarks with Multi-Scalar Multiplication

7.3.2 Pairing Protocols

7.3.3 Use-Case Case Study

To demonstrate the practical impact of our optimizations, we compared our approach against alternative systems for the common use case of verifying credential expiration. Table 15 presents the results for generating and verifying a proof that a credential has not expired.

Table 14. Performance of Anonymous Credential Operations (time in ms), n is attribute count

n	[ASM06]	[CDL16]	[PS16]	[TBA$^+$22] **??**	Our Improved **??**
			Obtain		
2	0.51	0.90	0.66	0.25	**0.23**
5	0.65	1.00	0.66	0.28	**0.27**
10	0.67	1.13	0.82	0.36	**0.31**
15	0.78	1.26	0.87	0.37	**0.36**
20	0.86	1.38	0.94	**0.41**	**0.41**
30	1.07	1.63	1.11	0.51	**0.49**
			Issue		
2	1.25	**0.72**	1.48	1.27	2.99
5	1.66	**0.75**	1.79	1.66	3.31
10	2.33	**0.83**	2.54	2.35	4.00
15	2.98	**0.84**	3.23	3.03	4.64
20	3.96	**0.90**	3.79	3.66	5.88
30	4.97	**0.94**	5.16	5.10	6.86
			Show		
2	5.39	2.31	3.20	**1.14**	1.29
5	6.05	2.42	3.15	**1.16**	1.29
10	7.44	1.71	4.53	**1.22**	1.33
15	8.86	2.71	6.14	1.40	**1.37**
20	11.88	1.88	7.66	**1.41**	1.51
30	12.91	3.15	16.23	**1.37**	1.59
			Verify		
2	7.59	2.18	4.57	2.47	**1.79**
5	9.25	2.25	5.52	2.73	**2.01**
10	11.09	**2.25**	7.10	3.16	2.44
15	13.96	**2.30**	8.62	3.47	2.72
20	16.93	**2.34**	9.88	3.84	3.21
30	26.30	**2.57**	16.55	4.67	3.79
			Issuing Phase Total (Obtain + Issue)		
2	1.76	1.62	2.14	**1.53**	3.22
5	2.31	**1.76**	2.45	1.95	3.57
10	3.00	**1.96**	3.37	2.71	4.31
15	3.75	**2.10**	4.10	3.40	5.00
20	4.82	**2.29**	4.74	4.06	6.28
30	6.04	**2.57**	6.27	5.60	7.35
			Showing Phase Total (Show + Verify)		
2	12.98	4.48	7.77	3.61	**3.08**
5	15.30	4.67	8.68	3.90	**3.30**
10	18.53	3.96	11.62	4.38	**3.77**
15	22.82	4.22	14.76	4.87	**4.09**
20	28.81	5.01	17.53	5.25	**4.72**
30	39.21	5.72	32.77	6.04	**5.37**

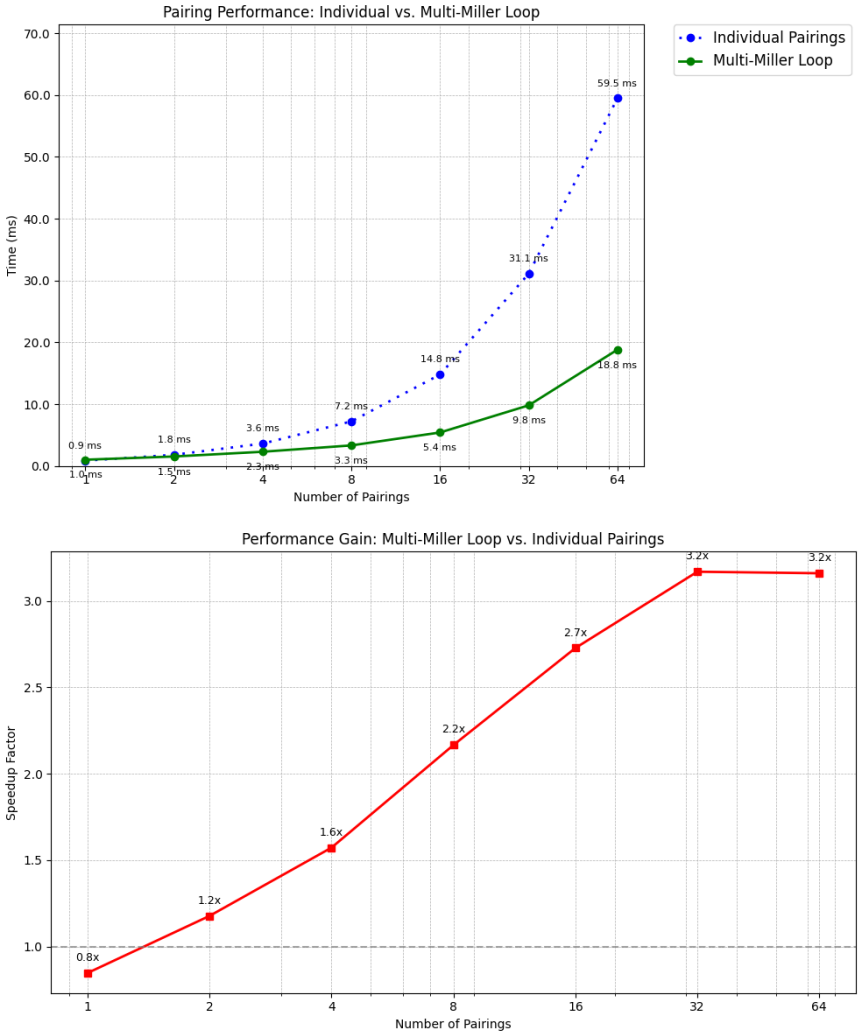

Fig. 22. Elliptic Curve Pairings - Practical Benchmarks with Miller-Loop Intermediate Computation

Table 15. Performance Comparison for Credential Operations

Approach	**Scheme**	**Show (ms)**	**Verify (ms)**	**Proof Size**
Simple Possession	[RWGM22]	2	2	424B
	Us	2	2	
Expiry	[RWGM22]	2	2	424B
	Us	2	2	
Linkable Show	[RWGM22]	41		
	Us	2	2	
Rate Limiting	[RWGM22]	58		
	Us	2	2	

Our evaluation reveals that our system can verify a credential's expiry status in just 3.3ms (combined Show+Verify), compared to 45-55ms for ZK-Creds approaches—a performance improvement of over 13×. This dramatic difference highlights how our optimized signature scheme and sigma protocol enables efficient predicate verification without sacrificing privacy.

Importantly, while general-purpose zero-knowledge systems provide flexibility, they introduce substantial computational overhead that makes interactive verification scenarios impractical. Our approach achieves similar expressiveness with dramatically better performance for the most common credential verification operations.

References

AAM. AAMVA. Jurisdiction Data Maps - American Association of Motor Vehicle Administrators - AAMVA.

ABB+18. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–15, Porto Portugal, April 2018. ACM.

AJ13. G. Alpár and B. Jacobs. Towards Practical Attribute-Based Identity Management: The IRMA Trajectory. In *Policies and Research in Identity Management*, pages 1–3. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. Series Title: IFIP Advances in Information and Communication Technology.

ark22. arkworks contributors. arkworks zkSNARK ecosystem, 2022.

ASM06. M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. *Security and Cryptography for Networks*, 4116:111–125, 2006. Series Title: Lecture Notes in Computer Science.

AWS. AWS. Verifiable credentials (claims) for government - Government Lens.

BBS04. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *Advances in Cryptology – CRYPTO 2004*, pages 41–55, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

BCJ+24. F. Baldimtsi, K. K. Chalkias, Y. Ji, J. Lindstrøm, D. Maram, B. Riva, A. Roy, M. Sedaghat, and J. Wang. zkLogin: Privacy-Preserving Blockchain Authentication with Existing Credentials. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3182–3196, Salt Lake City UT USA, December 2024. ACM.

Bit20. N. Bitansky. Verifiable Random Functions from Non-interactive Witness-Indistinguishable Proofs. *Journal of Cryptology*, 33(2):459–493, April 2020.

BLS01. D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

Bra00. S. A. Brands. *Rethinking public key infrastructures and digital certificates: building in privacy.* MIT Press, Cambridge, Mass, 2000.

BRS23. F. Benhamouda, M. Raykova, and K. Seth. Anonymous Counting Tokens. In *Advances in Cryptology – ASIACRYPT 2023*, pages 245–278. Springer Nature Singapore, Singapore, 2023. Series Title: Lecture Notes in Computer Science.

BSCG+14. E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, San Jose, CA, May 2014. IEEE.

C2P24. C2PA.ORG. Content Credentials : C2PA Technical Specification :: C2PA Specifications, 2024.

CDH+25. S. Celi, A. Davidson, H. Haddadi, G. Pestana, and J. Rowell. DiStefano: Decentralized Infrastructure for Sharing Trusted Encrypted Facts and Nothing More. In *Proceedings 2025 Network and Distributed System Security Symposium*, San Diego, CA, USA, 2025. Internet Society.

CDL16. J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited, 2016. Publication info: Published elsewhere. Major revision. Trust and Trustworthy Computing 2016.

CDS94. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Advances in Cryptology — CRYPTO '94*, pages 174–187, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

Cha81. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

CKL+16. J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. O. Pedersen. Formal Treatment of Privacy-Enhancing Credential Systems. In *Selected Areas in Cryptography – SAC 2015*, pages 3–24. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science.

CKS24. E. Crites, A. Kiayias, and A. Sarencheh. SyRA: Sybil-Resilient Anonymous Signatures with Applications to Decentralized Identity, 2024. Publication info: Preprint.

CL02. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology — CRYPTO 2002*, pages 61–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

CL03. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. *Security in Communication Networks*, 2576:268–289, 2003. Series Title: Lecture Notes in Computer Science.

CL04. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. *Advances in Cryptology – CRYPTO 2004*, 3152:56–72, 2004. Series Title: Lecture Notes in Computer Science.

CLPK22. A. Connolly, P. Lafourcade, and O. Perez Kempner. Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes. In *Public-Key Cryptography – PKC 2022*, pages 409–438, Cham, 2022. Springer International Publishing. Series Title: Lecture Notes in Computer Science.

CVH02. J. Camenisch and E. Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, Washington, DC USA, November 2002. ACM.

Dam10. I. Damgård. Sigma, 2010.

DCMMH22. P. Dzurenda, R. Casanova-Marqués, L. Malina, and J. Hajny. Real-world Deployment of Privacy-Enhancing Authentication System using Attribute-based Credentials. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–9, Vienna Austria, August 2022. ACM.

DGK+20. I. Damgård, C. Ganesh, H. Khoshakhlagh, C. Orlandi, and L. Siniscalchi. Balancing Privacy and Accountability in Blockchain Identity Management, 2020. Publication info: Published elsewhere. CT-RSA Conference.

Doc. Dock Labs. Dock Labs - Decentralized ID Management.

DY05. Y. Dodis and A. Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography - PKC 2005*, pages 416–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. Series Title: Lecture Notes in Computer Science.

Eur24. European Parliament. MEPs to approve new scheme for an EU-wide digital wallet | 26-02-2024 | News | European Parliament, February 2024.

FHS19. G. Fuchsbauer, C. Hanser, and D. Slamanig. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

FS86. A. Fiat and A. Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

GG22. A. Gupta and K. Gurkan. PLUME: An ECDSA Nullifier Scheme for Unique Pseudonymity within Zero Knowledge Proofs, 2022.

GKB20. K. Gurkan, Koh Wei Jie, and Barry Whitehat. Community Proposal: Semaphore: Zero-Knowledge Signaling on Ethereum, March 2020.

GNP+15. S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. NSEC5: Provably Preventing DNSSEC Zone Enumeration. In *Proceedings 2015 Network and Distributed System Security Symposium*, San Diego, CA, 2015. Internet Society.

JML24. S. Jaques, H. Montgomery, and M. Lodder. ALLOSAUR: Accumulator with Low-Latency Oblivious Sublinear Anonymous credential Updates with Revocations. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '24, pages 1708–1723, New York, NY, USA, July 2024. Association for Computing Machinery.

Mic. Microsoft. Microsoft Entra Verified ID | Microsoft Security.

MMZ+20. D. Maram, H. Malvai, F. Zhang, N. Jean-Louis, A. Frolov, T. Kell, T. Lobban, C. Moy, A. Juels, and A. Miller. CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability, 2020. Publication info: Published elsewhere. 2021 IEEE Symposium on Security and Privacy.

MRV99. S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 120–130, New York City, NY, USA, 1999. IEEE Comput. Soc.

MSK02. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 85(2):481–484, 2002. Publisher: The Institute of Electronics, Information and Communication Engineers.

noa24. Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework, April 2024. Doc ID: 32024R1183 Doc Sector: 3 Doc Title: Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework Doc Type: R Usr_lan: en.

Pol25. S. Polgar. Anonymous Credentials ecosystem, https://github.com/sampolgar/anonymous-credentials, 2025.

PS16. D. Pointcheval and O. Sanders. Short Randomizable Signatures. *Topics in Cryptology - CT-RSA 2016*, 9610:111–126, 2016. Series Title: Lecture Notes in Computer Science.

RAR+24. R. Rabaninejad, B. Abdolmaleki, S. Ramacher, D. Slamanig, and A. Michalas. Attribute-Based Threshold Issuance Anonymous Counting Tokens and Its Application to Sybil-Resistant Self-Sovereign Identity. *IACR Cryptol. ePrint Arch.*, page 1024, 2024.

RWG+18. H. Ritzdorf, K. Wust, A. Gervais, G. Felley, and S. Capkun. TLS-N: Non-repudiation over TLS Enabling Ubiquitous Content Signing. In *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018. Internet Society.

RWGM22. M. Rosenberg, J. White, C. Garman, and I. Miers. zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure, 2022. Publication info: Published elsewhere. Major revision. 2023 IEEE Symposium on Security and Privacy (SP).

SABB+20. A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers, March 2020. arXiv:1802.07344 [cs].

TBA+22. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. UTT: Decentralized Ecash with Accountable Privacy. 2022.

VB22. G. Vitto and A. Biryukov. Dynamic Universal Accumulator with Batch Update over Bilinear Groups. In *Topics in Cryptology – CT-RSA 2022*, pages 395–426. Springer International Publishing, Cham, 2022. Series Title: Lecture Notes in Computer Science.

VS16. F. Veseli and J. Serna. Evaluation of Privacy-ABC Technologies - a Study on the Computational Efficiency. In *Trust Management X*, pages 63–78. Springer International Publishing, Cham, 2016. Series Title: IFIP Advances in Information and Communication Technology.

WGD+25. W3C, Grant Noble, Dave Longley, Daniel C. Burnett, Brent Zundel, and Kyle Den Hartog. Verifiable Credentials Data Model v2.0, March 2025.

WMD+22. W3C, Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized Identifiers (DIDs) v1.0, July 2022.

ZKS+21. Z. Zhang, M. Król, A. Sonnino, L. Zhang, and E. Rivière. EL PASSO: Privacy-preserving, Asynchronous Single Sign-On. *Proceedings on Privacy Enhancing Technologies*, 2021(2):70–87, April 2021. arXiv:2002.10289 [cs].

ZMM+20. F. Zhang, S. K. D. Maram, H. Malvai, S. Goldfeder, and A. Juels. DECO: Liberating Web Data Using Decentralized Oracles for TLS. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1919–1938, October 2020. arXiv:1909.00938 [cs].