# Fast & Expressive Anonymous Credentials and Zero-Knowledge Proofs 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

# Abstract

Digital credential wallets manage identity documents such as government IDs and financial certificates, face the trilemma of privacy, security, and usability. Optimizing for anonymity by using Anonymous Credentials enhances privacy, but introduces challenges. Current benchmarks show verification using zero-knowledge proofs taking 50–500ms, far exceeding the $<$1ms of standard credentials, impeding usability. Additionally, anonymity complicates security: preventing multiple-credential issuance (sybil resistance) or enforcing revocation becomes difficult when both users and objects are essentially secret. These issues are urgent due to the EU's 2026 mandate for EU-wide credential wallet usage, which will drive widespread adoption of digital credential wallets, while critical use cases, like privately combining credentials from multiple issuers for KYC, emphasize the importance of this work.

This thesis extends existing work and develops new, fast cryptographic primitives for privacy-preserving credential wallets. It introduces *the fastest* anonymous credential scheme with a 3.77ms Show+Verify time for 10 attributes, outperforming prior methods by 10-15%. Three extensions enhance this scheme. 1) formalized Identity Binding property for secure multi-issuer, multi-credential verification, with an implementation verifying 16 credentials from unique issuers in 72ms; 2) new nullifier constructions using $\Sigma$-protocols without pairings, improving privacy-preserving sybil resistance by 5x over previous approaches; 3) T-SIRIS, a threshold-issued, sybil-resistant identity system with near-constant Show+Verify times, over 30x faster than comparable systems [RAR$^+$24]. These advancements are validated by an open-source Rust benchmarking library, delivering standardized empirical data across anonymous credential schemes.

A fast, feature-rich digital credential wallet using anonymous credentials benefits users and organizations. Users gain control of their data, enhancing privacy and reducing personal information exposure. Organizations, especially in banking and finance, can reduce application abandonment from slow or invasive verification. Verification times under 100ms—3.77ms for single credentials and 72ms for 16 simultaneously—make these primitives practical for real-world deployment without hindering user experience. This efficiency suits KYC/AML, age verification, and digital identity applications where speed and privacy are critical.

# Statement of Attribution

The work in this Thesis has not been published or submitted for publication in other formats as of the time of submission. I am the author of all chapters and the corresponding software packages.

*As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.*

*Qiang Tang*

# Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Samuel Polgar

# Acknowledgements

I thank my supervisor, A/Prof Qiang Tang, for accepting me as a student, for his curious nature, and for his confidence in me to explore my research topic. I'm grateful to my lab collaborators, Tian Qiu and Ya-Nan Li, for the fruitful discussions we had that enriched my understanding. I acknowledge the use of generative AI tools, which have proved valuable for ideation, troubleshooting, and code generation and testing. This research was supported by an Australian Government Research Training Program (RTP) Scholarship for my tuition fees.

I am grateful to my parents who have supported me through life's ups and downs and have always been there.

I thank my dog Zsuzsi for keeping me company during the 4 months I spent writing my Thesis at home.

I thank the Achilles running club for the weekly dose of positivity and my marathon running partner, Stephen Green, for constant inspiration; at 70 years old, vision impaired, and undergoing chemotherapy, he continues to run marathons.

Most importantly, I thank my wife, Marietta, who believed in me and selflessly convinced me to pursue my dream.

# Table of Contents

## List of Figures

# Chapter 1

# Introduction

## 1.1 The Rise of Digital Identity Wallets and the Need for Secure, Private, Usable Solutions

Digital credential wallets, the digital analogue of their physical counterparts, are emerging as essential tools for identity verification in both online and offline contexts. Nations such as India, Singapore, South Korea, Estonia, and Norway have implemented nationwide digital identity systems, while the EU mandates wallets and digital credential issuance across member states by 2026, and the US advances pilots in 13 states. These systems aim to streamline identity management by using interoperable wallets built on open standards, yet their success depends on resolving a fundamental trilemma: ensuring privacy, security, and usability. Consider the following scenarios

1. **KYC/AML Bank Application**: Opening a bank account requires verifying citizenship, age, government issued credentials, and possibly financial standing (bank balances). Traditional KYC processes are slow and complex, with up to 68% of users abandoning applications, with additional privacy risks of sharing and storing user personal information

2. **Social Media Verification:** Accessing social media in Australia will soon require age (over 16) and citizenship verification. A frequent transaction that could leak a childs online movements

3. **Age and Health Status:** Interacting with a medical clinic often requires different criteria to be met, such as a vaccination status, age, and insurance status. Current processes require a user to show all information during the process.

Across these cases, users seek privacy; disclosing only essential information, while verifiers require security, such as authenticity and fraud resistance. Usability demands *fast* verification of multiple, heterogeneous credentials. Yet, prior systems expose user data and fuel cybercrime, or privacy-preserving systems lack efficiency. The previous generation[1] of anonymous credential frameworks [CL04, ASM06, PS16], deployed in Microsoft's U-Prove and IBM's Idemix [CVH02, CKL+16], though innovative, met basic needs—verifying a single credential in 50ms—but fail to address the complex, multi-credential and sybil resistant demands of the next generation of identity systems. Verifying multiple credentials is either too slow, lacks a theoretical framework and security properties, or lacks efficient and private mechanisms for sybil resistance and revocation. These gaps are already driving discussions at the Internet Identity Workshop [Int25] and looking for solutions with discussions on how the credential wallet will securely issue and verify credentials from multiple different issuers (government, private, credential oracles); how can an efficient sybil resistance mechanism be incorporated inside a credential wallet.

This thesis develops cryptographic primitives to enable digital credential wallets that are fast, private, and resilient to abuse, paving the way for advanced identity solutions that solve for the upcoming limitations that credential wallets will pose, and unlock new functionality for secure and private digital interactions.

## 1.2 Anonymous Credentials

Anonymous credential systems enable privacy-preserving authentication and access control, allowing users to prove specific attributes without revealing their full identity. Anonymous Credentials is

[1] We term the "previous generation" as credentials which compute presentation proofs using $\mathbb{G}_T$ points, as this Thesis shows, is significantly slower. We term "new generation" credential frameworks that compute proofs over $\mathbb{G}_1$ [CDL16, TZ23, TBA+22], demonstrating remarkable efficiency improvements

the base building block of digital credential wallets, where users manage diverse credentials (e.g., passports, bank statements) to verify identity across scenarios like KYC or age verification. They fulfill two primary roles: ensuring holder authenticity like signature verification (e.g., proving knowledge of a secret key) and enforcing access control (e.g., demonstrating eligibility based on attributes like age $> 21$).

Anonymous Credentials are a subset of Verifiable Credentials, defined by W3C [WGD+25], as "tamper-evident credential whose authorship can be cryptographically verified", a Verifiable Presentation is a function of a VC which creates "A tamper-evident presentation of information encoded in such a way that authorship of the data can be trusted after a process of cryptographic verification." This Thesis focuses on private Verifiable Presentations that leverage zero-knowledge proofs to privately attest to the data within the Verifiable Credential.

The Anonymous Credential framework involves three main actors: the *user*, who owns the credential wallet with credentials, their attributes, and their signing/verification keys; the *issuer*, who creates credentials; and the *verifier*, who checks them. Credentials are authenticated data structures that cryptographically bind attributes, ensuring both hiding (privacy) and binding (security). These structures vary by issuance model and underlying cryptographic primitive. Credentials could be issued by a single-issuer as most are issued today, or to protect against malicious issuers, it could be threshold of issuers or even blockchain-based smart contracts. Credentials could be in the form of signatures or even data-structures in a cryptographic accumulator. The choice of cryptographic data structure dictates the proof system, the proof expressiveness, and the verification efficiency.

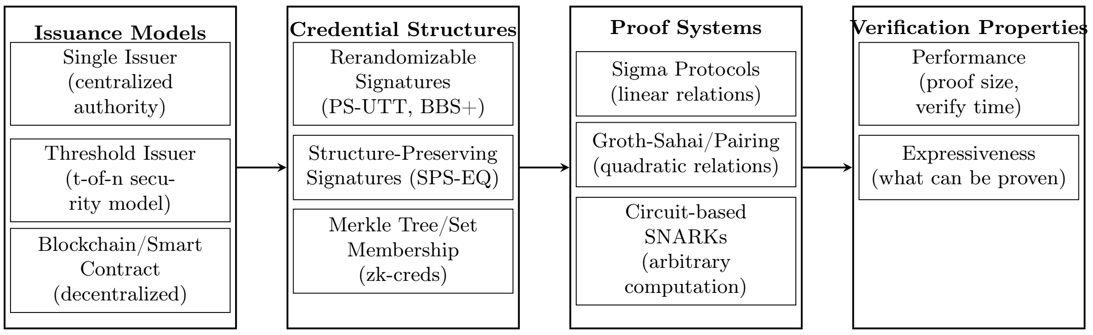
Issuance Models
Single Issuer (centralized authority)
Threshold Issuer (t-of-n security model)
Blockchain/Smart Contract (decentralized)
Credential Structures
Rerandomizable Signatures (PS-UTT, BBS+)
Structure-Preserving Signatures (SPS-EQ)
Merkle Tree/Set Membership (zk-creds)
Proof Systems
Sigma Protocols (linear relations)
Groth-Sahai/Pairing (quadratic relations)
Circuit-based SNARKs (arbitrary computation)
Verification Properties
Performance (proof size, verify time)
Expressiveness (what can be proven)

**Fig. 1.** Anonymous Credential Framework. The Issuance Model determines the Credential Type. The Credential Structure and underlying commitment determines the Proof System and Verification Properties

Two core interactive processes govern anonymous credentials. In the *obtain/issue* phase, users commit attributes, and issuers sign them, often blindly to preserve a user's privacy. In the *show/verify* phase, users generate a Verifiable Presentation to meet dynamic verification requirements (e.g., proving age $> 16$ without disclosing birthdate). Verifiers check these proofs for validity. Proof systems, such as Sigma protocols, Groth-Sahai, pairing-based methods or zk-SNARKS, balance expressiveness (e.g., proving complex predicates) with efficiency.

Security hinges on two main properties: *unforgeability*, ensured by the binding nature of commitments and soundness of proofs, and *anonymity*, achieved through hiding commitments, homomorphism of randomizing actions and the zero-knowledge property of zero-knowledge proofs, preventing linkability across verifications. However, designing systems that scale to multi-credential wallets, resist abuse, and perform efficiently involves trade-offs in privacy, efficiency, and expressiveness. These challenges highlight the need for careful selection of combining the fastest cryptographic primitives or optimizing the existing primitives to minimize computational overhead, selecting the most expressive schemes to enable the use-case functionality that is needed for next-generation identity systems. This Thesis addresses each of these and demonstrates our state-of-the-art primitives via qualitative and quantitiave analysis.

## 1.3 Limitations of Previous Approaches

The Trilemma between Privacy, Security, and Usability presents ongoing challenges and limitations. Privacy-preserving systems aim for *Accountable Privacy* within acceptable usability limits. The Trilemma tension arises due to the paradoxical nature of each property. Increasing Privacy and Security comes at the cost of Usability, and vice versa. The Gap Analysis often articulates the *cost of privacy* system architects must weigh when considering privacy-preserving primitives.

### 1.3.1 Limitation 1: Fast Anonymous Credentials (close to ECDSA)

Anonymous credentials fundamentally operate slower than traditional signature schemes like ECDSA which has long limited their adoption. Benchmarks [VS16] have shown $\mathsf{Show} + \mathsf{Verify}$ time for Anonymous Credential systems IBM's Idemix to be $(110ms, 220ms, 450ms)$ for 1,2,3 credentials, Microsoft's U-Prove to be $(180ms, 460ms, 600ms)$ for 1,2,3 credentials respectively, IRMA [AJ13] at $1300ms$ and zk-creds at $465ms$ [RWGM22]. Consider Alice approaching a turnstyle for the train needing to verify her transport ticket and student status. Non-private signatures would sign and verify under $1ms$, whereas current anonymous credentials benchmarks are beyond the $100ms$ rule applied by user experience experts [Jak09] and threshold for revenue loss due to webpage loads at Google and Amazon [Lin06]. The consequence is users and system architects prioritizing efficiency over privacy and using non-private methods of authentication.

### 1.3.2 Limitation 2: Security Against Malicious Issuers

Most Anonymous Credential systems deployed today assume honest-but-curious issuers rather than actively malicious ones; such systems claim strong security and privacy guarantees but fail to recognize the threat from maliciously generated signing and verification keys, which could allow a signer to secretly embed tracking or de-anonymizing algebraic structure into the credential. This would undermine the systems privacy guarantees and prevent widespread adoption.

### 1.3.3 Limitation 3: Efficient Expressive Proofs

Current Anonymous Credential Benchmarks show computationally expensive expressive proof evaluation. [VS16] demonstrates Idemix running Show+Verify for a credential with 5 attributes including string equality (120ms) and separately "date greater than" (310ms). zk-creds [RWGM22] demonstrates two use-cases, accessing a website proving over 18 with Show+Verify of 148ms best case or 602ms worst case, and verifying a credential in-person and selectively disclosing age and photo being 103ms best case and 602ms worst case[2]. All of which are over the $100ms$ threshold discussed in Limitation 1 1.3.1, hindering realistic uptake of new technology.

### 1.3.4 Limitation 4: Secure and Efficient Verification of Multiple Credentials

There has been some consideration of the need to securely verify multiple credentials simultaneously while proving that such credentials come from the same user. Credential wallets provide the perfect use-case, our use-cases 1.1 are prototypical but it is easy to see where this functionality can be extended to, online application KYC requirements show 68% of consumers abandon an their application because of delays [Sig22], especially with content credentials [C2P24] producing a credential for each transaction, photo, or message sent from a device. [CKL$^+$16] includes the functionality in security proofs, but not as an explicit property; furthermore, while it is a theoretical property, no schemes attempt to clearly present benchmarks on practical feasibility.

### 1.3.5 Limitation 5: Efficient Nullifiers for Sybil Resistance

Nullifiers are used in privacy-preserving systems to prove sybil resistance while retaining anonymity. A user generates a nullifier from their key and some input and wants to keep their key secret to uphold anonymity (and prevent tracking through system use). In payment systems, the input might be a

[2] worst case scenarios come from Merkle Tree proof update

coin's serial number [BSCG+14, TBA+22], and Identity Systems might use the credential context, e.g., "passport" to ensure a user has only 1 passport, or "2025vote" to enforce sybil-resistant voting. Zerocash [BSCG+14] and PLUME [GG22] leverage zk-SNARKS ($50 + ms$) while [TBA+22] leverage pairing constructions ($12.38ms$), all with computational overhead (possibly more than one nullifier can be used in a presentation), which combined with credential verification and expressive proofs will extend Show+Verify times beyond acceptable limits, rendering the cost of privacy too high for practical use.

### 1.3.6 Limitation 6: Efficient Threshold Signatures for Threshold Issued Identity System

Current threshold credential schemes for threshold-issued identity systems present with prohibitively slow operations or lack benchmark analysis to conclude otherwise. Threshold BBS+ was proposed [DKL+23] and initial benchmarks show signing times of 38 seconds [HNY+25] though the scheme is without identifiable abort, its inclusion would be more expensive. tACT-based systems like S3ID [RAR+24] show verification times that scale poorly with attribute count. Threshold implementations of SPS-EQ [CKP+23] may address efficiency, but they lack performance benchmarks. The performance gap, lack of evaluation metrics create and concrete implementations in an identity system create barriers to adoption.

### 1.3.7 Limitation 7: Understanding the (Empirical) Cost of Privacy

Organizations wanting to adopt Anonymous Credential frameworks for integration to their identity systems lack empirical benchmarks across various schemes which would allow them to select a scheme based on its properties and empirical analysis. It's hard to understand the "cost" of privacy based on the literature and implementations available. Furthermore, IT departments benefit from experimenting with code to learn. Some academic implementations include code samples but often are built with different libraries, languages, and enhancements - it's difficult to assess and compare schemes in this environment.

## 1.4 Contributions

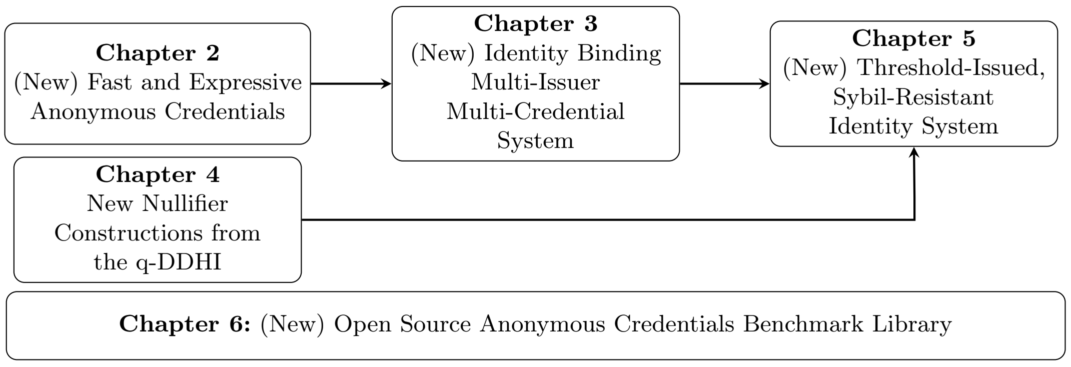
**Chapter 2**
(New) Fast and Expressive Anonymous Credentials
**Chapter 3**
(New) Identity Binding Multi-Issuer Multi-Credential System
**Chapter 5**
(New) Threshold-Issued, Sybil-Resistant Identity System
**Chapter 4**
New Nullifier Constructions from the q-DDHI
**Chapter 6:** (New) Open Source Anonymous Credentials Benchmark Library

**Fig. 2.** Contribution Roadmap

### 1.4.1 Chapter 2: Fast and Expressive Anonymous Credentials from New Rerandomizable Signatures

This chapter directly targets the security-privacy-usability trilemma by improving the base building block, the credential construction itself. I address **limitations 1, 2, and 3**, by constructing the most efficient Anonymous Credential for Verifiable Presentations (according to my research and benchmarks) by enhancing a variant [TBA+22] of the PS signature [PS16] with Show+Verify time of 3.77ms for a credential with 10 attributes. I also provide empirical, fair benchmarks against the state-of-the-art schemes [ASM06,CDL16,PS16,TBA+22] using the same cryptography libraries and coding language to ensure comparison takes into account the cryptography itself. Additionally, I included security against malicious issuers by adapting the framework in [FHS19], and proved my construction of Rerandomizable Signatures secure in the Anonymous Credential model from [FHS19]. I also demonstrate that the proof system, $\Sigma$-protocols, excels in satisfying the efficiency-expressive tradeoff.

### 1.4.2 Chapter 3: Identity Binding Multi Issuer Multi Credential Anonymous Credentials

I address **limitation 4** in this chapter by tackling the challenge of verifying multiple credentials from different issuers while proving they belong to the same user, a key requirement for credential wallets. I create the *Identity Binding* security property for such schemes. I address **limitation 7** by extensively benchmarking the use-case scenarios. I built a Multi-Issuer Multi-Credential System in Rust and benchmarked different scenarios. First, *non-private* multi-credential usage for 16 unique credentials, 16 attributes each credential from unique issuers, cost just  27ms (Verify). The *cost of privacy* for (Show + Verify) was just 2.6x at  72ms. I also show that credentials signed by a single issuer with our construction can use signature aggregation and reduce (Show + Verify) to just 31ms for the same (16 credentials with 16 attributes in each). This chapter proves that complex identity assertion requirements like KYC/AML, incorporating multiple credentials, can be extraordinarily efficient with credential wallets.

### 1.4.3 Chapter 4: New Nullifier Constructions from the q-DDHI and Applications to Accountably Private Systems.

In this chapter we resolve **limitation 5** by introducing fast, private nullifier constructions using the $q-$DDHI assumption and novel $\Sigma$-protocols. I first introduce a pairing-free (Non-Private) Dodis Yampolskiy (DY) [DY05] Verifiable Random Function (VRF) construction from the $q-$DDHI assumption (rather than $q$-DBDHI) that is 3x more efficient than the original, and because we remove pairings, we can use non-pairing curves e.g. Ed25519 for 6x speedup. This has applications broader than nullifier schemes. Next, I developed 3 x novel $\Sigma-$ protocols to privately prove the DY structure (in zero knowledge) from committed attributes and constructed deterministic and probabilistic nullifiers with applications to sybil resistance and revocation in privacy-preserving protocols. The deterministic nullifier is at least 5x faster than previous constructions [TBA$^+$22].

### 1.4.4 Chapter 5: Threshold-Issued, Sybil-Resistant Private Identity System from Anonymous Credentials and Nullifiers

This chapter constructs a Threshold-Issued, Sybil-Resistant identity system addressing **limitation 6**. We first demonstrate our base-building block, the Threshold Signature construction is extremely efficient as we compare against tACT a state-of-the-art system [RAR$^+$24]. We then construct a Sybil Resistant Identity System drawing on the work in Chapters 2, 3, 4 and show it's exponentially faster for end to end performance metrics 5.3.

### 1.4.5 Chapter 6: Open Source Anonymous Credential Library

To bridge the gap in standardized evaluations, we develop an open-source library benchmarking state-of-the-art credential schemes under consistent conditions. This resolves **limitation 7**, giving practitioners the tools to test different constructions and work with empirical data. This library also reveals practical efficiency gains from cryptography library functions like Multi-Scalar Multiplication, Miller-Loop Pairing computation.

# Chapter 2

# Fast and Expressive Anonymous Credentials from New Rerandomizable Signatures

### 2.0.6 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.

Previous deployments of Anonymous Credentials 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 I 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.

### 2.0.7 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 I'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 I'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. I 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:** I provide formal security proofs that were only sketched in [TBA+22]:
   - Position-binding vector commitments under SDLP in the Algebraic Group Model (Section 2.1.2)
   - EUF-CMA security for rerandomizable signatures over commitments under PS-LRSW (Section 2.2)
   - Satisfaction of Anonymous Credential model [FHS19] security properties—correctness, unforgeability, and anonymity—for multi-show operations (Section 2.5)
2. **Fastest Show+Verify Operations:** My construction achieves 10-16% faster verification than UTT [TBA+22] and outperforms the most popular BBS+ variant [CDL16] - I validate through comprehensive benchmarks 7.
3. **Malicious Issuer Security through Key Verification Protocols:** I formalize anonymity guarantees against malicious issuer via mandatory key verification (Section 2.3.2), preventing deanonymization attacks through maliciously generated keys—a vulnerability that existing schemes do not address.
4. **Empirical Validation of Schnorr Protocol Efficiency:** Comprehensive benchmarks 22 reveal 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** Section 2.1 establishes the background preliminaries, section 2.2 provides the first complete security proofs for our rerandomizable signature scheme. Section 2.3 presents my 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 2.4 demonstrates how Sigma protocols achieve efficient predicate evaluation over complex policies. Section 2.5 constructs my complete credential system with security proofs. Section 2.6 validates my approach through benchmarks, showing state-of-the-art performance with up to 83% verification speedup.

## 2.1 Preliminaries

### 2.1.1 Assumptions

**Definition 2.1 (Symmetric Discrete Logarithm Assumption (SDLP)).** *For any PPT adversary $\mathcal{A}$, we say the SDLP assumption holds if there exists a negligible function* $\mathsf{negl}$ *such that:*

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

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

**Definition 2.2 (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* $\mathsf{negl}$ *such that:*

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

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

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

### 2.1.2 Dual-Group Pedersen Commitments

In this section, I introduce a specialized extension of Pedersen commitments that supports vector messages, rerandomizability, and position binding. My 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 2.3 (Commitment Scheme).** *A commitment scheme* $\mathsf{Com}$ *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 $\lambda$ and outputs a commitment key* $\mathsf{ck}$*. The message space $\mathcal{M}$ is implicitly defined by* $\mathsf{ck}$.

– $\mathsf{Commit}(\mathsf{ck}, m) \to (\mathsf{cm}, r)$*: is a probabilistic algorithm that takes as input the commitment key* $\mathsf{ck}$ *and a message $m \in \mathcal{M}$. It outputs a commitment* $\mathsf{cm}$ *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* $\mathsf{ck}$*, a commitment* $\mathsf{cm}$*, 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 2.4 (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 2.5 (Hiding).** *A commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is hiding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}$ *such that:*

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

**Definition 2.6 (Binding).** *A commitment scheme* (Setup, Commit, Open) *is binding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

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

**Dual-Group Construction for Vector of Messages** I 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 the 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 $\mathsf{ck}$ 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.

### 2.1.3 Rerandomizable Signature over Commitments

**Definition 2.7 (Signature Scheme).** *A signature scheme* Sig *is a tuple of probabilistic polynomial-time algorithms* (KeyGen, Sign, Verify) *where:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$*: is a probabilistic algorithm that takes as input a security parameter $\lambda$ 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 $\sigma$.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to b$*: is a deterministic algorithm that takes as input the public key* pk*, a message $m \in \mathcal{M}$, and a signature $\sigma$. It outputs a bit $b \in \{0, 1\}$, where 1 indicates acceptance and 0 indicates rejection.*

**Definition 2.8 (Correctness).** *A signature scheme* (KeyGen, Sign, 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 2.9 (EUF-CMA Security).** *A signature scheme* (KeyGen, Sign, 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 2.10 (Rerandomizable Signature).** *A rerandomizable signature scheme over commitments* $\mathsf{RS}$ *extends a standard signature scheme with the following interface:*

– $\mathsf{KeyGen}(1^\lambda, \mathsf{ck}) \to (\mathsf{sk}, \mathsf{pk})$*: Takes security parameter* $\lambda$ *and commitment key* $\mathsf{ck}$*, outputs signing key* $\mathsf{sk}$ *and verification key* $\mathsf{pk}$*.*

– $\mathsf{Sign}(\mathsf{sk}, \mathsf{cm}; u) \to \sigma$*: Signs commitment* $\mathsf{cm}$ *using signing key* $\mathsf{sk}$ *and randomness* $u$*.*

– $\mathsf{Rerand}(\mathsf{pk}, \sigma, \Delta_r, \Delta_u) \to \sigma'$*: Creates a rerandomized signature* $\sigma'$ *from signature* $\sigma$ *using randomization values* $\Delta_r, \Delta_u$*.*

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

**Definition 2.11 (Correctness).** *A rerandomizable signature scheme over commitments satisfies correctness if for all security parameters* $\lambda$*, 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* $u, \Delta_r, \Delta_u$*:*

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 2.12 (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* $\mathsf{negl}$ *such that:*

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

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

### 2.1.4 Zero Knowledge Proofs and Sigma-protocols

#### Proof Relations

**Definition 2.13 (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 2.14 (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* $\kappa$ *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 2.15 (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 2.16 (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 2.17 (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 $a = g^z h^{-e}$.

## 2.2 Formal Treatment for UTT's Rerandomizable Signature Over Commitments

I build upon the rerandomizable signature scheme over commitments introduced in UTT [TBA$^+$22]. This signature scheme enables the anonymous credential system to present signatures that are both unlinkable and unforgeable without revealing the underlying identity attributes. My main contribution is the first complete and tight security proof in the Algebraic Group Model that establishes $\mathsf{EUF\text{-}CMA}$ security with minimal assumptions (noted that [TBA$^+$22] presents a proof sketch).

### 2.2.1 Position Binding Pedersen Commitments for a Vector of Messages

Building on the standard commitment scheme defined in Section 2.3, I extend it with the following properties:

**Definition 2.18 (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* $\mathsf{negl}$ *such that:*

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

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

**Definition 2.19 (Rerandomizability).** *A commitment scheme* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ *is rerandomizable if it includes an additional algorithm* $\mathsf{Rerand}$ *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* $\mathsf{ck}$*, a commitment* $\mathsf{cm}$*, and additional randomness $\Delta_r$. It outputs a new commitment* $\mathsf{cm}'$ *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 $\Delta_r$:*

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

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

### 2.2.2 SDLP and Position Binding Security Analysis in the AGM

The 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, $\mathsf{cm}$ and $\widetilde{\mathsf{cm}}$ are uniformly distributed in $\mathbb{G}_1$ and $\mathbb{G}_2$, revealing no information about $\mathsf{attrs}$.

– **Position Binding:** Under the SDLP assumption in the AGM, the scheme is position binding. The reduction 2.20 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 $\mathsf{Rerand}$ 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 $\mathsf{attrs}$. 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 2.20.** *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 $\mathsf{ck}$ 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 $\mathsf{ck}$.**

   – 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 $\mathsf{ck}$:

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

   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 $\mathsf{cm}$ opening to both $(\mathsf{attrs}_0, r_0)$ and $(\mathsf{attrs}_1, r_1)$.

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 $\mathsf{cm}$: $g_k = g^{y_k}$ 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 $\epsilon$, then $\mathcal{B}$ solves SDLP with probability $\epsilon/n$.

**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.

### 2.2.3 EUF-CMA Rerandomizable Signature over Commitments

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

– $\mathsf{CM.Setup}(1^\lambda, \ell, (y_i, \ldots, y_\ell \in \mathbb{Z}_p^\ell)) \rightarrow \mathsf{ck}$ : 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 $\mathsf{ck} \leftarrow (g, (g_1, \ldots, g_\ell), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_\ell))$

– $\mathsf{RS.KeyGen}(1^\lambda, \mathsf{ck}) \rightarrow (\mathsf{sk}, \mathsf{vk})$ : Retrieve $(g, \cdot, \tilde{g}, \cdot)$ from $\mathsf{ck}$, Sample $x \leftarrow\$ \mathbb{Z}_p$, Set $(\mathsf{sk}, \mathsf{vk}) \leftarrow (g^x, \tilde{g}^x)$, return $(\mathsf{sk}, \mathsf{vk}))$

– $\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{cm}; u) \rightarrow \sigma$ : Let $h \leftarrow g^u$ Return $\sigma \leftarrow (h, (\mathsf{sk} \cdot \mathsf{cm})^u)$

– $\mathsf{RS.Rerand}(\sigma, \Delta_r, \Delta_u) \rightarrow \sigma'$ : Parse $\sigma$ as $(\sigma_1, \sigma_2)$ 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 $\sigma$ as $(\sigma_1, \sigma_2)$, The prover $\mathcal{P}$ runs a Proof of Knowledge protocol with the following relation

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

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

### 2.2.4 Security Analysis in the AGM

**Theorem 2.21.** *The rerandomizable signature scheme is correct.*

*Proof.* First we demonstrate the prover's rerandomized signature verifies with the verification key $\mathsf{vk}$ 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 2.1, 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 2.22 (EUF-CMA Security).** *Assume the PS-LRSW assumption holds and the Pedersen commitment is computationally binding. Then, in the Algebraic Group Model, the 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 $\epsilon$.

**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 $\mathsf{pk}$ 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 $\sigma_1, \sigma_2$ 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 $\sigma_1^*$ 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* My 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.

## 2.3 Extending UTT's Construction

### 2.3.1 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:

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

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 $\widetilde{\sigma} = (\widetilde{\sigma}_1, \widetilde{\sigma}_2) \in \mathbb{G}_2$ 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 2.6.2).

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

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

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 2.6.2). 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.

### 2.3.2 Malicious Issuer Security Guarnatees

My 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 $\mathsf{ck}$. Issuer parses $\mathsf{ck}$ as $\{g^{y_i}, \tilde{g}^{y_i}\}_{i\in[\ell]}$ and runs a $\Sigma$-protocol 2.3.2 to prove the correctness of $\mathsf{vk}$ and $\mathsf{ck}$

**Protocol 1: Proving Correctness of Issuer Keys**

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

**Prover Input:** $x, y_1, \dots, y_\ell \in \mathbb{Z}_q$ 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 $r_x \leftarrow\$ \mathbb{Z}_q$, $r_1, \dots, r_n \leftarrow\$ \mathbb{Z}_q$ 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** I separate my security analysis of the verification key $\mathsf{vk}$ and commitment key $\mathsf{ck}$ for clarity:

**Theorem 2.23.** *The $\Sigma$-protocol for* $\mathsf{RS.VerKey}$*, defined in Protocol 2.3.2, 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 2.24.** *The $\Sigma$-protocol for* $\mathsf{RS.VerKey}$*, defined in Protocol 2.3.2, 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 $\mathsf{ck}_i$ 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:

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

For the pairing check, we need to show:

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

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.

## 2.4 Sigma Protocols for Efficient, Expressive Proofs

The algebraic structure of an Anonymous Credential scheme determines the compatible proof system. In this work, I focus on $\Sigma-$protocols as the proof system of choice due to their efficiency and sufficient expressiveness for practical credential use cases. Sigma protocols are three-move, interactive zero-knowledge proofs over cyclic groups, such as $\mathbb{G}_1, \mathbb{G}_T$, where a prover demonstrates knowledge of a secret (e.g., an exponent $x$)) satisfying a specific relation, under the discrete logarithm assumption. These protocols are particularly well-suited for proving linear relations over scalars, which are common in certain credential types.

The credential construction builds on the line of work following initial rerandomizable signatures on bilinear groups, such as CL04 [CL04], BBS+ [ASM06,CDL16,TZ23], and PS [PS16,TBA$^+$22]. Earlier schemes [CL04,ASM06,PS16] required proofs over exponents in $\mathbb{G}_T$, with sub-optimal computational overhead. In contrast, newer constructions [CDL16,TZ23,TBA$^+$22] restructure the algebraic setup to enable zero-knowledge proofs over $\mathbb{G}_1$ points, minimizing computation by leveraging faster scalar exponent operations and efficient optimizations like Multi-Scalar Multiplication.

In these schemes, the prover uses $\Sigma$-protocols to prove knowledge of exponents (scalars) corresponding to hidden attributes. The soundness of the security proofs relies on the extraction of these scalars, ensuring unforgeability. $\Sigma$-protocols excel in efficiency for such tasks, with basic operations like credential verification, selective disclosure, and equality proofs achieving show+verify times of 0.49–0.98 ms (see Table 1). Furthermore, sigma protocols naturally support composition for more expressive statements, such as AND/OR combinations and linear relations, with minimal overhead (approximately 2–3 times the baseline cost). They can also be extended to handle range proofs [BBB+17] and set membership proofs via accumulators [CL02, JML24], making them versatile for credential systems.

**Predicate Satisfaction** In my credential system, a predicate $\phi$ is a boolean function over an attribute vector $\mathsf{attrs}$, formally $\phi : \mathcal{A} \to \{0,1\}$, where $\mathcal{A}$ is the space of attribute vectors. For a credential with attributes $\mathsf{attrs} = [\mathrm{id}, \mathrm{k}, \mathrm{ctx}, \mathrm{exp}]$, we say that "$\mathsf{attrs}$ satisfies $\phi$" if $\phi(\mathbf{a}) = 1$.

The prover uses sigma protocols to demonstrate that their attributes $\mathsf{attrs}$ satisfy a given predicate $\phi$ (i.e., $\phi(\mathsf{attrs}) = 1$) in zero-knowledge, meaning the verifier learns only that the predicate holds, not the attribute values. For example, consider a predicate $\phi_{\text{age}}$ : age $> 18$. The prover proves knowledge of attributes $\mathsf{attrs} = [\text{age}, \dots]$ such that age $> 18$, without revealing the exact value of age.

The soundness of the $\Sigma$-protocol ensures that a prover cannot generate a valid proof for $\phi(\mathsf{attrs}) = 1$ unless their attributes truly satisfy the predicate, reinforcing the unforgeability of my credential system.

**Table 1.** Sigma Protocol Efficiency Summary

| **Operation** | **Native?** | **Core Operations** | **Overhead** | **Prove (ms)** | **Verify (ms)** |
|---|---|---|---|---|---|
| Schnorr (baseline) | Yes | Basic commitment opening | 1× | 0.38 | 0.49 |
| AND (2 statements) | Yes | Parallel openings, shared challenge | 2× | 0.76 | 0.98 |
| OR (2 options) | Yes | 2 commitments, 1 real + 1 simulated | 2× | 0.76 | 0.98 |
| NAND | Yes-ish | AND/OR combination | 2-3× | 0.76-1.14 | 0.98-1.47 |
| Equality | Yes | Single opening with special structure | 1-1.2× | 0.38-0.46 | 0.49-0.59 |
| Linear Relation | Yes | Modified response computation | 1-1.2× | 0.38-0.46 | 0.49-0.59 |
| Greater/Less or Eq | No | Extended sigma with log. complexity | 20-30× | 8-15 | 8-15 |
| Range Proof | No | Extended sigma with bulletproofs | 20-30× | 8-15 | 8-15 |
| Set Membership (OR) | No | OR across all set elements | N× | 0.38 × N | 0.49 × N |
| Set Membership (Acc) | No | Sigma proof of witness knowledge | 10-15× | 4-6 | 4-6 |

- **Overhead:** Relative cost compared to basic Schnorr protocol
- **Times:** Based on MSM-optimized implementation with 16-attribute credential
- **Native:** Protocols directly expressible with basic sigma operations vs. requiring extended techniques
- **N:** For Set Membership (OR), N = size of the set

### 2.4.1 Groth-Sahai Proofs

Groth-Sahai [GS08, EG14] proofs underpin Structure-Preserving Signatures [KPW15, FHS19, CLPK22, CKP+23] where the verification keys, messages, and signatures are group elements. GS proofs offer more expressive proofs for complex statements than $\Sigma$-protocols such as quadratic relations using Pairing-Product Equations (PPE). They enable commitments to group elements with soundness relying on extracting the elements under falsifiable assumptions e.g. SXDH. However, expressiveness comes at a cost, each committed attribute may require 4x $\mathbb{G}_1$ + 4x$\mathbb{G}_2$ per committed attribute [KPW15, SFA23] and multiple pairings for verification, quickly increasing verification time exponentially over $\Sigma$-protocols. A more detailed analysis is found 2.

### 2.4.2 zk-SNARKs (Groth16)

zk-SNARKs provide maximimum expressiveness, supporting proofs of any NP statement. In the zk-creds system [RWGM22] (along the line of work starting with [GGM14] ) they prove set membership in merkle trees as the credential verification, their hash is a non-algebraic structure incompatible with $\Sigma$-protocols and GS proofs. Groth16 [Gro16] proofs offer constant-size proofs ($2x\mathbb{G}_1, +\mathbb{G}_2$) and fast verification. Yet their proving computational is preventative for wide-scale adoption in credential systems requiring fast Show+Verify algorithms, costing 465 ms(worst case) for a simple credential show in zk-creds, compared to 3.77ms for credential show for my signature and $\Sigma$-protocols. Composing statements (e.g., AND operations) adds 43 ms just for the AND operation without the additional statement, whereas sigma protocols handle composition "for free" via shared challenges. Furthermore, such proofs do not allow multi-credential linking. Additionally, zk-SNARKs require a trusted setup, introducing a security risk absent in sigma protocols and Groth-Sahai proofs, and rely on non-falsifiable assumptions. A detailed analysis is found 2.

**Table 2.** Comparative Analysis of Zero-Knowledge Proof Systems

| **Aspect** | **Sigma Protocols** | **Groth-Sahai Proofs** | **zk-SNARKs (Groth16)** |
|---|---|---|---|
| **Expressiveness** | Linear relations | Quadratic relations in bilinear groups | General arithmetic circuits |
| **Operations** | Discrete log proofs, commitment openings, equality | Pairing product equations | Any NP statement |
| **Proof Size** | Linear in statement size | $O(N)$ with $4\lvert G_1\rvert + 4\lvert G_2\rvert$ per committed variable | Constant: $2\lvert G_1\rvert + \lvert G_2\rvert$ |
| **Proving Cost** | 0.38-0.76ms for basic operations | Multiple exponentiations + 8 elements per equation | $O(n)$ circuit operations |
| **Verification Cost** | 0.49-0.98ms for basic operations | 4 verification equations with multiple pairings per statement | Constant (3 pairings) |
| **Trusted Setup** | No | No | Yes |
| **Assumptions** | Discrete log | Falsifiable (SXDH, standard) | Non-falsifiable |
| **Complexity** | Low | Medium | High |
| **Use Cases** | Credential verification, selective disclosure | Structure-preserving signatures, pairing-based schemes | Applications requiring minimal proof size |

$\Sigma$-protocols outperform Groth-Sahai proofs and zk-SNARKs in verification efficiency (0.49–0.98 ms vs. higher costs due to pairings or circuit complexity) and avoid trusted setups. Their expressiveness, while limited to linear relations, suffices for credential operations like selective disclosure and equality proofs, as shown in Table 1. Groth-Sahai proofs excel in pairing-based schemes but suffer from large proof sizes and slow verification for large credentials. zk-SNARKs, despite their versatility and compact proofs, are hindered by slow proving times (e.g., 465 ms in zk-creds) and setup requirements, making them less practical where efficiency is paramount.

## 2.5 Expressive ABC from Rerandomizeable Signatures

I construct an Attribute-Based Anonymous Credential System from the 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.

### 2.5.1 Syntax

**Definition 2.25 (ABC System).** *An Attribute-based Anonymous Credential system consists of the following probabilistic polynomial-time (PPT) algorithms:*

– $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{ppar})$*: Takes a security parameter* $\lambda$ *and outputs public parameters* $\mathsf{ppar}$*.*

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

### 2.5.2 Security Model

The 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**

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

**Oracles**

- $\mathcal{O}_{\mathsf{HU}}()$: Creates a new honest user $i$, adds them to $\mathsf{HU}$, and returns $i$
- $\mathcal{O}_{\mathsf{CU}}(i)$: Corrupts user $i$ by moving them from $\mathsf{HU}$ to $\mathsf{CU}$, revealing their secret keys and all credentials owned by $i$
- $\mathcal{O}_{\mathsf{Obtain}}(i, \vec{m})$: Issues a credential $\mathsf{cred}$ to user $i$ for the attribute vector $\vec{m}$, provided $i \in \mathsf{HU}$. The credential is added to $\mathsf{CRED}$, and $\mathsf{OWNR}[\mathsf{cred}]$ is set to $i$
- $\mathcal{O}_{\mathsf{Show}}(i, \phi)$: Generates a proof $\pi$ that the credentials of user $i$ satisfy the predicate $\phi$, provided $i \in \mathsf{HU}$ and the credentials meet the condition $\phi$

**Definition 2.26 (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.

```
$\mathrm{Game}^{\mathsf{UNF}}_{\mathsf{ABC},\mathcal{A}}(1^\lambda)$
    // Challenger Setup
Initialize $\mathsf{HU} \leftarrow \emptyset, \mathsf{CU} \leftarrow \emptyset,$
$\mathsf{CRED} \leftarrow \emptyset, \mathsf{OWNR} \leftarrow \{\}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
    // $\mathcal{A}$ queries oracles
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk})$
    // Forgery
$\mathcal{A}$ outputs $(\mathsf{cred}'^* = (\sigma'^*, \mathsf{cm}'^*), \phi^*, \pi^*)$
    // Winning Condition
$\mathsf{Verify}(\mathsf{cred}'^*, \phi^*, \pi^*, \mathsf{opk}) = 1 \wedge$
$\mathsf{OWNR}[\mathsf{cred}'^*] \notin \mathsf{CU} \;\vee\; \phi^*(\vec{m}^*) = 0$
    // i.e., either the credential belongs to an honest user
    // or does not satisfy the claimed predicate
```

```
$\mathrm{Game}^{\mathsf{ANON}}_{\mathsf{ABC},\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{HU} \leftarrow \emptyset, \mathsf{CU} \leftarrow \emptyset$   // Challenger Setup
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathcal{A}(\mathsf{OrgKeyGen}(\mathsf{pp}))$   // Adversary may corrupt issuer
For $i \in \{0, 1\}$:   // Setup two challenge users
    $\mathsf{r}_i \leftarrow \mathsf{UserKeyGen}(\mathsf{pp}), \mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
    $\vec{m}_i \leftarrow \mathcal{A}(i)$ such that $\phi(\vec{m}_i) = 1$
    $\mathsf{cm}_i \leftarrow \mathsf{CM.Com}(\vec{m}_i; \mathsf{r}_i), \mathsf{cred}_i \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm}_i)$
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk})$   // Learning Phase
$\phi \leftarrow \mathcal{A}()$   // Challenge Phase
Assert $\phi(\vec{m}_0) = \phi(\vec{m}_1) = 1$ and $\{0, 1\} \subset \mathsf{HU}$
$b \leftarrow\!\!\$ \{0, 1\}$   // Challenger samples random bit
$(\mathsf{cred}', \mathsf{cm}', \pi) \leftarrow \mathsf{Show}(\mathsf{cred}_b, \mathsf{cm}_b, \mathsf{r}_b, \phi)$
$b' \leftarrow \mathcal{A}(\mathsf{cred}', \mathsf{cm}', \pi)$   // Adversary guesses
return $b'$, // Return the adversary guess
```

**Fig. 3.** Unforgeability and Anonymity Games for the ABC System

```
$\mathcal{O}_{\mathsf{HU}}()$
if $i \notin \mathsf{HU} \cup \mathsf{CU}$
    $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
if $i \in \mathsf{HU}$ :
    $\mathsf{HU} \leftarrow \mathsf{HU} \setminus \{i\}$
    $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$
    $\mathsf{creds}_i \leftarrow \{\mathsf{cred} | \mathsf{OWNR}[\mathsf{cred}] = i\}$
    return $\{(\mathsf{cred}, \mathsf{r}) | (\mathsf{cred}, \mathsf{cm}, \vec{m}, \mathsf{r}, i) \in \mathsf{CRED}\}$
return $\perp$
```

```
$\mathcal{O}_{\mathsf{Obtain}}(i, \vec{m})$
if $i \in \mathsf{HU}$ :
    $\mathsf{r} \leftarrow\!\!\$ \mathbb{Z}_p$
    $\mathsf{cm} \leftarrow \mathsf{CM.Com}([\vec{m}]; \mathsf{r})$
    $\mathsf{cred} \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm})$
    $\mathsf{CRED} \leftarrow \mathsf{CRED} \cup \{(\mathsf{cred}, \mathsf{cm}, \vec{m}, \mathsf{r}, i)\}$
    $\mathsf{OWNR}[\mathsf{cred}] = i$
return cred

$\mathcal{O}_{\mathsf{Show}}(i, \phi)$
if $i \in \mathsf{HU} \;\wedge\; \phi(\mathsf{cred}_i) = 1$ :
    Parse $\mathsf{cred}_i = (\sigma, \mathsf{cm}, \vec{m}, \mathsf{r})$
    $\pi \leftarrow \mathsf{Show}(\mathsf{cred}_i, \phi)$
    $\mathsf{SHOW} \leftarrow \mathsf{SHOW} \cup \{(i, \phi, \pi)\}$
    return $\pi$
return $\perp$
```

**Fig. 4.** Oracles for the ABC Security Games

**Definition 2.27 (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.

**Definition 2.28 (Anonymity).** *An* ABC *system provides anonymity if, for all PPT adversaries $\mathcal{A}$, the advantage in the following experiment is negligible:*

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

*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.

### 2.5.3 My Construction

**Outline** My credential system operates over attribute space $\mathbb{Z}_p$. The user is indexed by $i$. The credential $\mathsf{cred}$ 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 $\Sigma$-protocols to prove their correctness for any predicate $\phi$. 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 6.3.1 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 $\mathsf{Obtain}, \mathsf{Issue}$, the user sends the commitment $\mathsf{cm}$ along with a proof of opening $\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm})$ allowing the extraction of $\mathsf{r}$ for corrupt users in the unforgeability proof.

### 2.5.4 The Sigma Protocols

My ABC system relies on five core relations proven via $\Sigma$-protocols:

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

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

4. **Malicious Issuer Protection:** For verification key $\mathsf{vk}$ and commitment key $\mathsf{ck}$, $\Pi^{\mathcal{R}_{\mathsf{verkey}}}$ 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 $\phi$ such that $\mathsf{ctx} =$ "*master*" $\wedge$ $\mathsf{exp} >$ today.

Passport

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

**Fig. 5.** 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 $\mathsf{ZKPoK}$ satisfying

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

**Freshness** We prevent replay attacks via the challenge phase of the $\Sigma$-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:

– $\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) \rightarrow \{0, 1\}$: 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.

### 2.5.5 Security of my Construction

**Theorem 2.29 (Unforgeability).** *The* $\mathsf{ABC}$ *system is unforgeable if the rerandomizable signature scheme is* $\mathsf{EUF\text{-}CMA}$ *secure, the Pedersen Commitment scheme is position binding, and the* $\Sigma$*−protocol is sound.*

*Proof (Sketch).* I reduce the security of my $\mathsf{ABC}$ system to three underlying properties: $\mathsf{EUF\text{-}CMA}$ security of the signature scheme, position binding of the commitment scheme, and the soundness of the $\Sigma$-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 $\mathsf{EUF\text{-}CMA}$ 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 $\mathsf{cred}^*$ contains a signature on a commitment not previously queried, $\mathcal{B}$ outputs this as an EUF-CMA forgery.
- **Case 2:** If $\mathsf{cred}^*$ contains a signature on a legitimate commitment but proves a predicate $\phi^*$ 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 $\pi^*$.
  - This witness must include attribute values that satisfy $\phi^*$.
  - Since the original attributes don't satisfy $\phi^*$, 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 $\pi^*$ verifies but neither Case 1 nor Case 2 applies, this directly contradicts the soundness of the Sigma protocol.

**Theorem 2.30 (Anonymity).** *The* $\mathsf{ABC}$ *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* $\Sigma$*-protocol 2.1.4 ensures a simulator generates* $\pi$ *indistinguishable from real proofs.*

*Proof (Sketch).* I prove anonymity using a hybrid argument that shows the adversary's advantage is negligible. The proof specifically addresses protection against malicious issuers through the 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 $\mathsf{Show}$ protocol.

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

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

<u>$\mathsf{OrgKeyGen}(1^\lambda, 1^\ell)$</u> for attribute vector length $\ell$

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

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

<u>$(\mathsf{Obtain}, \mathsf{Issue})$</u>:

$\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1) = \mathsf{ZKPoK}\{(\mathsf{s}_1, \mathsf{r}) \mid \mathsf{cm} = g_1^0 g_2^0 g_3^0 g_4^{\mathsf{s}_1} g^{\mathsf{r}}\}$
$\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) = \mathsf{ZKPoK}\{(\mathsf{sk}, x, \{y_i\}_{i=1}^{\ell}) \mid \mathsf{sk} = g^x \wedge \mathsf{vk} = \tilde{g}^x$
$\quad \bigwedge_{i=1}^{\ell} (g_i = g^{y_i} \wedge \tilde{g}_i = \tilde{g}^{y_i})\}$

| $\mathsf{Obtain}(\mathsf{opk})$ | | $\mathsf{Issue}(\Pi^{\mathcal{R}_{\mathsf{zero}}}, \mathsf{cm}, \mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{osk})$ |
|---|---|---|
| | $\xleftarrow{\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck})}$ | 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 $\bot$ | | |
| $\mathsf{s}_1 \leftarrow\$ \mathbb{Z}_p, \mathsf{r} \leftarrow\$ \mathbb{Z}_p$,<br>$\mathsf{cm}_1 = \mathsf{CM.Com}([0, 0, 0, \mathsf{s}_1]; \mathsf{r})$ | | |
| | $\xrightarrow{\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)}$ | If $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)$ fails, return $\bot$ |
| | $\mathsf{s}_2 \leftarrow\$ \mathbb{Z}_p, \mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}, \mathsf{exp}, \mathsf{s}_2]; 0)$ | where $\mathsf{ctx} =$ `"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 $\bot$ | $\xleftarrow{\sigma, \mathsf{cm}, \mathsf{s}_2, \mathsf{id}, \mathsf{ctx}, \mathsf{exp}}$ | $u \leftarrow\$ \mathbb{Z}_p,\ \sigma \leftarrow\$ \mathsf{RS.Sign}(\mathsf{cm}, \mathsf{osk}, u)$ |
| Else, return $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{r}, \mathsf{opk})$ | | |

<u>$(\mathsf{Show}, \mathsf{Verify})$</u> for credential $\mathsf{cred}$ and predicate $\phi$:

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

| $\mathsf{Show}(\mathsf{cred})$ | | $\mathsf{Verify}(\sigma', \mathsf{cm}', \pi_\phi, \mathsf{opk})$ |
|---|---|---|
| | | Send empty access policy $\phi = \bot$ |
| Parse $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{r}, \mathsf{opk})$<br>Sample $\Delta_{\mathsf{r}}, \Delta_u \leftarrow\$ \mathbb{Z}_p$<br>$\sigma' = \mathsf{RS.Rand}(\sigma, \Delta_{\mathsf{r}}, \Delta_u)$<br>$\mathsf{cm}' = \mathsf{CM.Rand}(\mathsf{cm}, \Delta_{\mathsf{r}})$,<br>$\mathsf{r}' = \mathsf{r} + \Delta_{\mathsf{r}}$<br>Compute $\Pi_\phi$ | | |
| | $\xrightarrow{\sigma', \mathsf{cm}', \pi_\phi}$ | If $\pi_\phi$ fails, return 0, else 1 |

**Fig. 6.** ABC System

**Indistinguishability of Hybrid 0 and Hybrid 1:** By the zero-knowledge property of the $\Sigma$-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 $\mathsf{Hybrid}_0$ and $\mathsf{Hybrid}_1$ is negligible.

**Independence of $b$ in Hybrid 1:** In $\mathsf{Hybrid}_1$, the adversary's view is independent of the bit $b$. This follows from:

1. **Perfect Hiding of Commitments:** The rerandomized commitment $\mathsf{cm}'$ 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 $\mathsf{cred}'$ 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 $\pi$ 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 $\mathsf{Hybrid}_1$, 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 $\mathsf{Hybrid}_0$ and $\mathsf{Hybrid}_1$, which is negligible under our security assumptions.

## 2.6 Performance Evaluation

I implemented all credential schemes using the arkworks library [ark22] in Rust, ensuring consistent cryptographic primitives across all implementations. My 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%. I 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, I leveraged arkworks' optimized MSM implementation that uses Pippenger's algorithm with windowing optimizations. All operations are single-threaded.

### 2.6.1 Empirically Faster Credential Presentations Compared to Original Construction in UTT

My $\mathbb{G}_2$ signature variant from 2.3.1 optimizes credential presentation over credential issuance, representing optimization of 10 - 15% improvement over the original approach [TBA+22] as seen in figure 7 and detailed in table 3. It's worth noting that a 10 - 15% improvement compounds when a user is verifying multiple credentials from a credential wallet. By relocating signature components from $\mathbb{G}_1$ to $\mathbb{G}_2$, I achieve faster credential presentation.

### 2.6.2 Empirically Faster Credential Presentations Compared to Popular Anonymous Credentials

I implemented and compared five Anonymous Credential schemes, BBS+ (2006) [ASM06]: The original BBS+ signature scheme, BBS+ (2016) [CDL16]: An optimized version with proofs in $\mathbb{G}_1$, PS (2016) [PS16]: The Pointcheval-Sanders scheme, UTT [TBA+22]: The UTT variant with signatures in $\mathbb{G}_1$ and my optimized variant with signatures in $\mathbb{G}_2$. I benchmarked the Anonymous Credential algorithms and present the data. Implementations and benchmarks are available for reproduction[3] [Pol25]

[3] https://github.com/sampolgar/anonymous-credentials

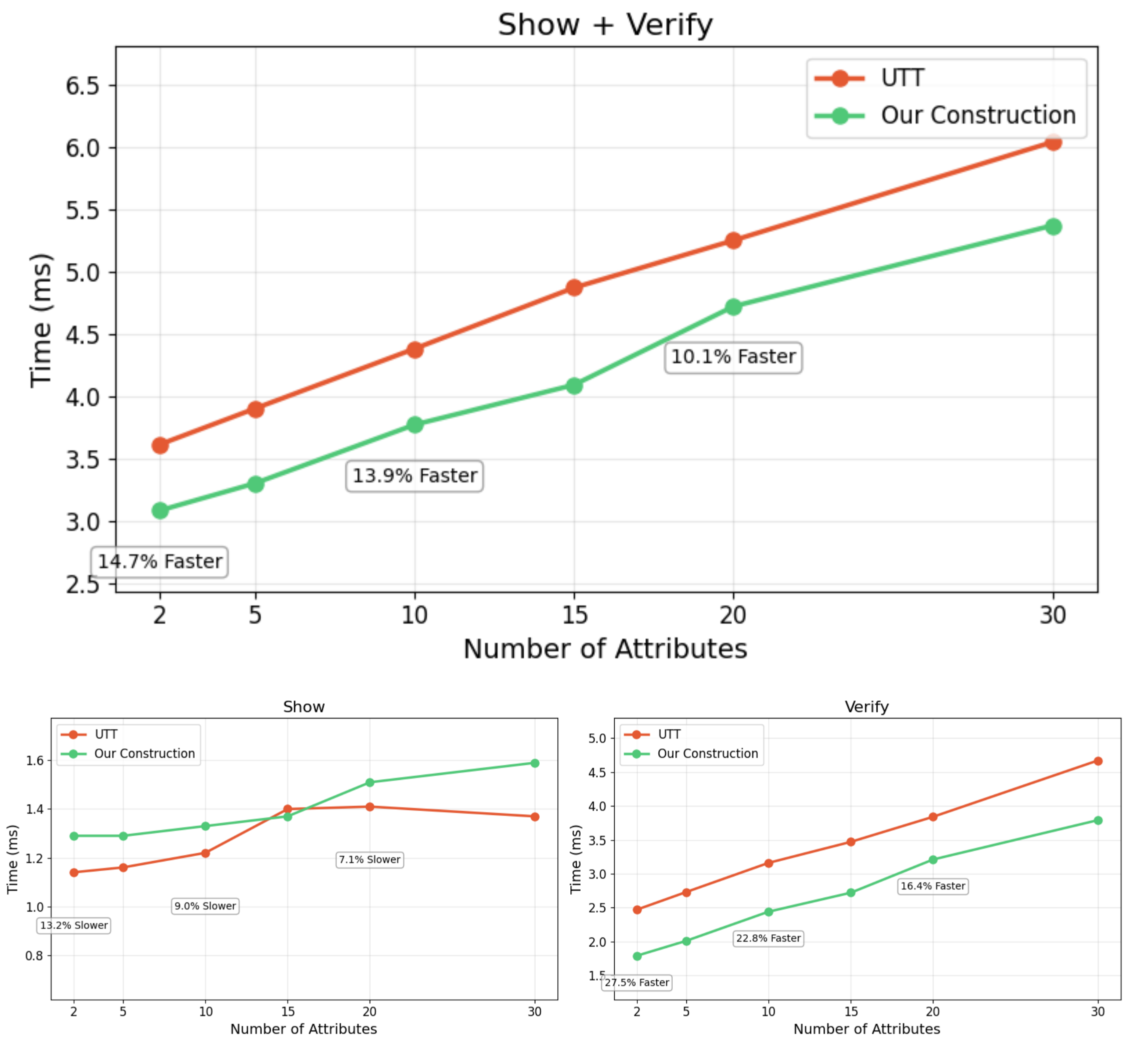

**Fig. 7.** Credential Presentation Comparison between UTT and my G2 variant - my construction is the fastest by 10 - 15%

**Table 3.** Performance Comparison: UTT and My Construction (time in ms)

| **Attributes** | **Show** | | **Verify** | | **Show + Verify** | |
|---|---|---|---|---|---|---|
| | **[TBA+22]** | **Mine** | **[TBA+22]** | **Mine** | **[TBA+22]** | **Mine** |
| 2 | 1.14 | 1.29 | 2.47 | 1.79 | 3.61 | 3.08 (14.7%↓) |
| 5 | 1.16 | 1.29 | 2.73 | 2.01 | 3.90 | 3.30 (15.4%↓) |
| 10 | 1.22 | 1.33 | 3.16 | 2.44 | 4.38 | 3.77 (13.9%↓) |
| 15 | 1.40 | 1.37 | 3.47 | 2.72 | 4.87 | 4.09 (16.0%↓) |
| 20 | 1.41 | 1.51 | 3.84 | 3.21 | 5.25 | 4.72 (10.1%↓) |
| 30 | 1.37 | 1.59 | 4.67 | 3.79 | 6.04 | 5.37 (11.1%↓) |

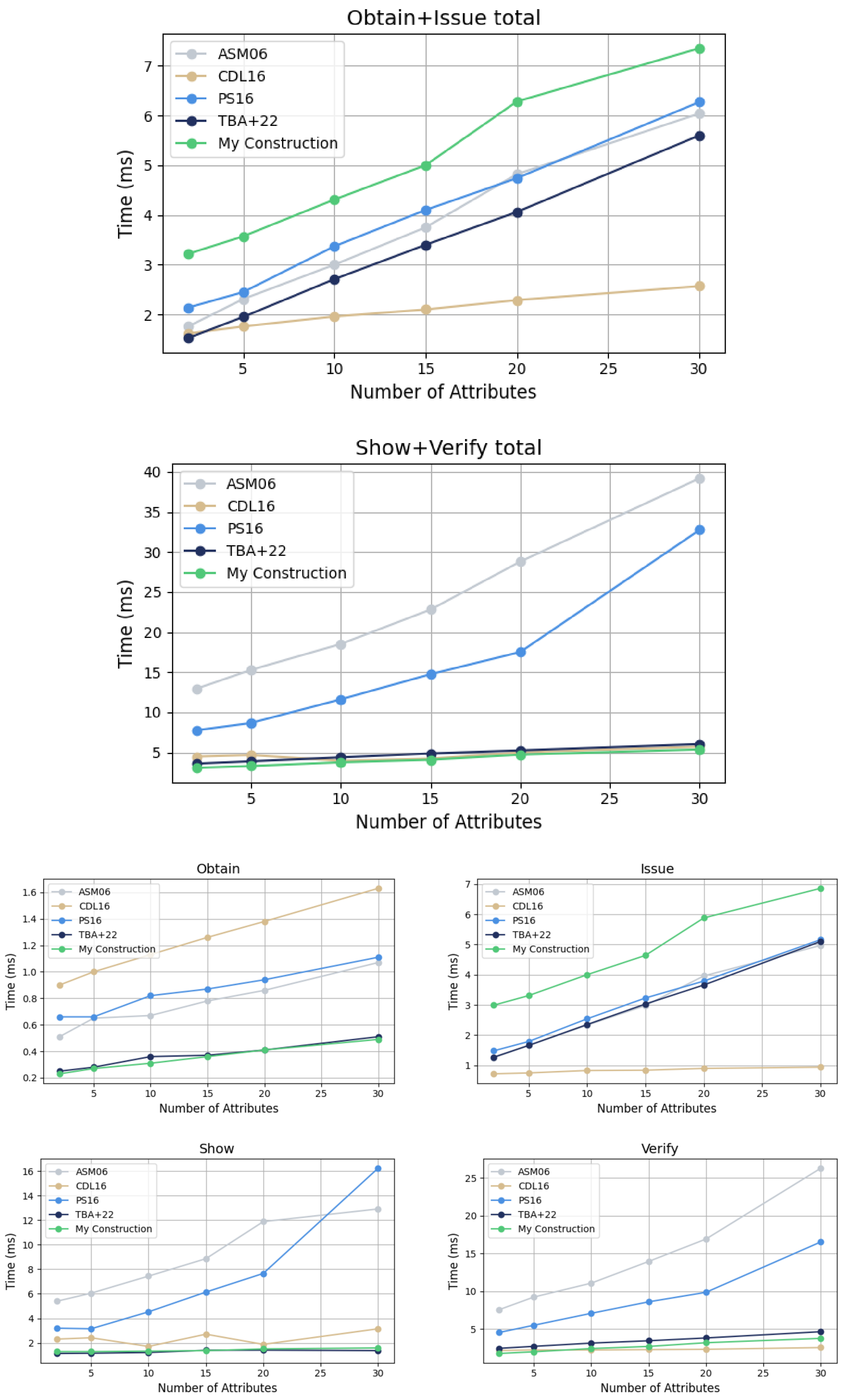

**Fig. 8.** Performance comparison of different operations across varying numbers of attributes

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

| $n$ | [ASM06] | [CDL16] | [PS16] | [TBA+22] | My Construction 2.3.1 |
|---|---|---|---|---|---|
| | | | **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** |

**Key Findings**

1. **Optimized**: My 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 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, My PS-UTT G2 is the most efficient.

## 2.7 Summary and Future Work

# Chapter 3

# Identity Binding Multi Issuer Multi Credential Anonymous Credentials

## 3.1 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 [HP22], and generic zkSNARKs, while expressive, incur high computational costs [RWGM22]. CanDID [MMZ+20] 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 $3\times$ slower than non-private baselines (e.g., 18.67ms vs. 6.77ms for 4 credentials), remains efficient for practical use. Implementations can be found on my GitHub[4]

### 3.1.1 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.

[4] https://github.com/sampolgar/mimc-abc

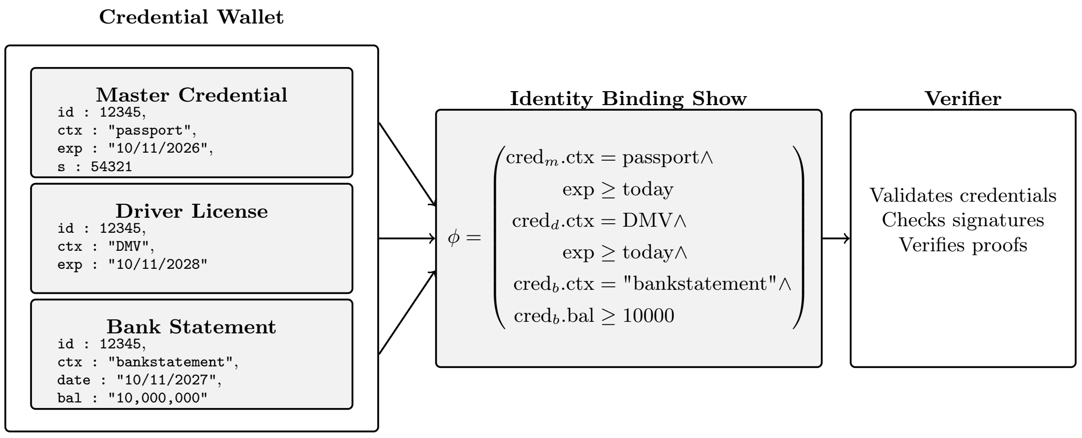

**Fig. 9.** A credential showing scheme for privacy-preserving protocols, illustrated with a credential hierarchy binding multiple credentials (passport, driver's license, bank statement) to enable secure and privacy-preserving presentation to verifiers.

## 3.2 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 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.

### 3.2.1 MIMC-ABC Syntax

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

**Definition 3.1 (MIMC-ABC).**

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: *Outputs public parameters* $\mathsf{pp}$, *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* $\mathsf{pp}$ *and $\ell$, 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* $\mathsf{id}$ *is a unique identifier,* $\mathsf{ctx}_j$ *is the issuer-specific context, and* $\mathsf{attrs}_j$ *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* $\mathsf{cm}$ *with* $\mathsf{osk}_j$, *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* $(\sigma_j, \mathsf{cm})$ *to* $(\sigma'_j, \mathsf{cm}')$, *and computes a proof $\pi$ that* $\phi(\{\mathsf{attrs}_j\}_j) = 1$ *and all* $\mathsf{id}$ *values match, using $\Sigma$-protocols. The verifier checks the proof and rerandomized credentials against* $\{\mathsf{opk}_j\}_j$, *outputting 1 if valid, 0 otherwise.*

### 3.2.2 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 $\phi$ 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 $\phi$ they cannot legitimately satisfy, including forging credentials or mixing credentials from different identities, except with negligible probability.

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

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

### 3.2.3 Identity Binding Property

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

**Definition 3.2 (Identity Binding).** *A MIMC-ABC system satisfies identity binding if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}$ such that:*

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

*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 $\mathsf{id}$, enforced via position-binding commitments and zero-knowledge proofs, as detailed in Section 3.

## 3.3 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 $\Sigma$-protocols to prove credential validity and identity consistency efficiently, supporting predicates over an arbitrary number of attributes.

### 3.3.1 Intuition

Each credential from issuer $j$ commits to an attribute vector $\mathsf{attrs}_j = [\mathsf{id}, \mathsf{ctx}_j, \mathsf{exp}_j]$, where $\mathsf{id}$ is a unique, user-chosen identifier, $\mathsf{ctx}_j$ denotes the credential's context (e.g., "passport"), and $\mathsf{exp}_j$ 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 $\mathsf{id}$ is a unique, user-chosen identifier, $\mathsf{ctx}_j$ denotes the credential's context (e.g., "passport"), and $\mathsf{exp}_j$ 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 $\mathsf{r}_j$ 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 $\mathsf{id}$, satisfying a predicate $\phi$.

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 10). 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 ($j = 3$) |
|---|---|---|
| id : `12345`, | id : `12345`, | 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. 10.** Example credentials from three issuers, sharing $\mathsf{id} = 12345$. Additional attributes (e.g., degree type) can be included.

### 3.3.2 Construction Details

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

– $\mathsf{Setup}(1^\lambda)$: Generates $\mathsf{pp}$ 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 $\mathsf{opk}_j = (\mathsf{vk}_j, \mathsf{ck}_j)$.

– $\mathsf{Obtain}$ and $\mathsf{Issue}$: 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 $\mathsf{r}_j$.

– $\mathsf{Show}$ and $\mathsf{Verify}$: 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 $\Sigma$-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 $\pi$ and $\{\mathsf{opk}_j\}_j$, accepting if valid.

### 3.3.3 Identity Binding Mechanism

Identity binding relies on the position-binding property of Pedersen commitments (Section 2.2). The $\Sigma$-protocol proves that all $\mathsf{cm}'_j$ share the same $\mathsf{id}$ in the first position ($g_1^{\mathsf{id}}$), 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 $\mathsf{id}$ values that appear equal, reducing security to standard cryptographic hardness.

## 3.4 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 $\Sigma$-protocols (Section 2.4.3).

### 3.4.1 Correctness

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

### 3.4.2 Unforgeability

**Theorem 3.3 (Unforgeability).** *MIMC-ABC is unforgeable if the rerandomizable signature scheme is EUF-CMA secure, the Pedersen commitment scheme is position-binding, and the $\Sigma$-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 $\mathsf{osk}_j$, we extract $\sigma'_j$ as an EUF-CMA forgery.

2. **Predicate Misuse**: If $\mathcal{A}$ uses valid credentials but proves a false $\phi^*$ (e.g., $\mathsf{exp}_j <$ today when $\mathsf{exp}_j >$ today), the $\Sigma$-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 $\mathsf{id}$ values yet proves same-$\mathsf{id}$, we extract two openings of some $\mathsf{cm}'_j$ (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 $\mathsf{cm}_j$. Any forgery violates one assumption, bounding $\mathcal{A}$'s advantage by $\mathsf{negl}(\lambda)$.

### 3.4.3 Anonymity

**Theorem 3.4 (Anonymity).** *MIMC-ABC is anonymous, even against colluding issuers, if the signature scheme's rerandomization is indistinguishable, the commitment scheme is hiding, and the $\Sigma$-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 $\mathsf{Show}$.

- **Hybrid 1**: Replace $\pi$ with a simulated proof using the $\Sigma$-protocol's zero-knowledge simulator.

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

### 3.4.4 Identity Binding

**Theorem 3.5 (Identity Binding).** *MIMC-ABC satisfies identity binding under the SDLP assumption (Section 2.1). For any PPT adversary $\mathcal{A}$ mixing credentials with distinct $\mathsf{id}$ 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 $\{\mathsf{cred}'_j\}_j$, $\phi^*$ requiring same $\mathsf{id}$, and $\pi^*$ verifying despite $\mathsf{id}_j \neq \mathsf{id}_k$ for some $j, k$, the $\Sigma$-protocol's special soundness extracts witnesses from $\pi^*$. 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 $\mathsf{id}_j$, and differing $\mathsf{id}$ values imply two openings of some $\mathsf{cm}'_j$ 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.

### 3.4.5 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.

## 3.5 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 2.3.1. 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.

### 3.5.1 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 $\mathsf{attrs}_j$ 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 $\Sigma$-protocol for same-$\mathsf{id}$ and predicate satisfaction.

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

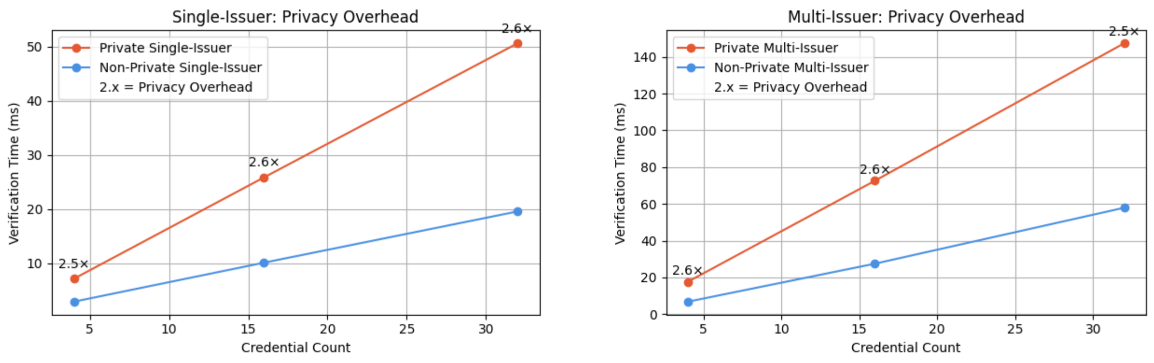

**Fig. 11.** These graphs show the cost of privacy for single-issuer (left) and multi-issuer(right) credential use with multiple credentials. The graph shows a 2.6x increase in Show+Verify time for private multi-credential use. The left graph shows single-issuer multi-credential verification (with batch verification). The right graph shows multi-issuer multi-credential verification (no batch verification). Both graphs use a credential with 16 attributes displaying Show+Verify time.

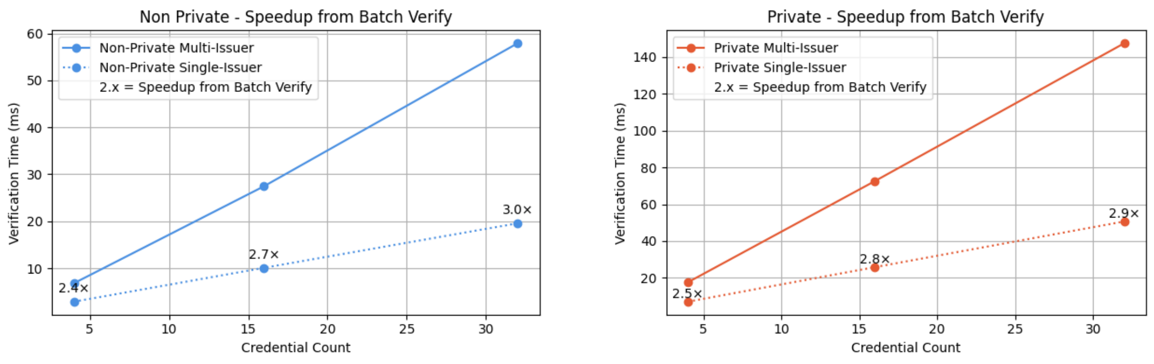

**Fig. 12.** These graphs show the benefit of single-issuer signature aggregation, but also show that multi-issuer credential verification adds small 2.4x-3x overhead and therefore does not impact the Show+Verify significantly

**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 |

### 3.5.2 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 $\Sigma$-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.

**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 |

# Chapter 4

# New Nullifiers and VRFs from the q-DDHI and Applications to Privacy Systems

By definition, an anonymous user's interaction with a system should be indistinguishable from any of its other interactions. Sybil resistance enforces identity uniqueness, preventing users from performing the same action more than once. The paradox of enforcing uniqueness of indistinguishable users is representative of the challenges of the security, privacy, and usability trilemma. A nullifier is a privacy-preserving, publicly-provable, unique output generated from a user's private information, enabling actions like spending coins, voting anonymously, or issuing credentials while preventing double-spending and maintaining anonymity through zero-knowledge proofs. Nullifiers are currently defined within specific contexts in privacy-preserving protocols like Zcash, Tornado Cash, and Crypto-Wallets like MetaMask and Ledger, but there is no standard definition with its own security properties. Current constructions either use computationally-heavy pairings and $\Sigma$-protocols [TBA$^+$22] taking 12.38ms, or support ECDSA leveraging zk-SNARK proofs [BSCG$^+$14, GG22], with between 10-45 seconds SNARK proof-generation overhead [Dis24]. My goal is to create the fastest nullifier scheme without relying on pairings or zk-SNARKS while ensuring compatibility with efficient $\Sigma$-protocols for composition within larger cryptographic protocols.

I start with a formal definition of a nullifier. I then present my nullifiers based on the $q-$DDHI assumption, which total computation (Eval, Prove, Verify) for both deterministic and probabilistic variants in 2.49ms and 3.66ms, respectively, on the BLS12-381 curve. Additional testing on secp256k1 shows 1.12ms and 1.88ms, Ed25519 shows 1.32ms and 2.34ms. Demonstrating my Nullifiers are at least 5x faster than the fastest Pairing-based construction [TBA$^+$22]. My constructions required 3 new $\Sigma$-protocols for proving a structure similar to the Dodis-Yampolskiy VRF structure. I also show a VRF construction from the $q-$DDHI assumption, which is 3x - 6x more efficient than the original. All implementations are available in my GitHub repository[5].

## 4.1 Chapter Roadmap and Contributions

1. **Fast DY VRF Variant:** I start with a detour in section 4.4 and construct a DY VRF in a prime-order group (without pairings) by creating a new $\Sigma$-protocol for *proving inverse linear relation* to directly verify the DY structure and show my construction is 3-5x efficiency improvement to the original DY, seen in figure 13.

2. **New Zero-Knowledge Proof of Knowledge of an Inverse Exponent:** The rest of the work focuses on privacy. My first step in section 4.4 is a new $\Sigma$-protocol to anonymously prove the DDHI challenge ($g^{1/x}$), which I generalise to a sigma protocol for *Zero-Knowledge Proof of Inverse Exponent.*

3. **Deterministic Nullifier:** In Section 4.5 I combine the *Zero Knowledge Proof of Knowledge of an Inverse Exponent* with the *Zero-Knowledge Proof of Inverse Linear Relation* to now anonymously prove the prime-order DY structure. I construct my deterministic and use this $\Sigma$,-protocol as the proof of correctness.

4. **Rerandomizable Nullifier:** I extend the Deterministic Nullifier protocol in a similar direction. I first extend the $\Sigma$-protocol to prove Zero-Knowledge Proof of Inverse Linear Relation within a

[5] https://github.com/sampolgar/nullifiers

commitment, that is $\mathsf{cm} = g_1^{1/s+x} g^r$ for randomly sampled $r$, I call this a Rerandomizable Nullifier as the output contains a fresh randomness $r$ each output.

## 4.2 Nullifier Definition

A nullifier Scheme takes as input a public commitment $\mathsf{cm}_s = \mathsf{CM.Com}([s])$to a private seed $s$, input $x$, optionally a nullifier list $\mathcal{N}$. It generates a pseudorandom deterministic nullifier $nf$, proof of correct evaluation $\Pi_{\mathsf{nf}}$, and optionally a proof of inclusion/exclusion within $\mathcal{NL}$.

**Definition 4.1 (Nullifier).** *A Nullifier Scheme is a tuple of PPT algorithms with input space $\mathcal{X}$, nullifier space $\mathcal{N}$ and proof space $\Pi$ where*

– $\mathsf{NF.Eval}(s, x) \to \mathsf{nf}$*: Computes the Nullifier output* $\mathsf{nf} \in \mathcal{N}$ *for input $x \in \mathcal{X}$ using seed $s$.*

– $\mathsf{NF.Prove}(\mathsf{cm}_s, x) \to \pi$*: Produces a proof $\pi \in \Pi$ that* $\mathsf{nf} = \mathsf{NF.Eval}(s, x)$ *is computed correctly from the public commitment to the private seed $s$.*

– $\mathsf{NF.Verify}(\mathsf{cm}, x, \mathsf{nf}, \pi) \to \{0, 1\}$*: Verifies that* $\mathsf{nf}$ *is the correct Nullifier output for input $x$, committed seed $s$ using proof $\pi$.*

A secure Nullifier scheme must satisfy the following properties:

– **Completeness**: For all seeds $s$, inputs $x \in \mathcal{X}$, and commitments $\mathsf{cm} = \mathsf{CM.Com}(s)$:

$$\Pr\left[\mathsf{NF.Verify}(\mathsf{cm}, x, \mathsf{nf}, \pi) = 1 : \begin{array}{c} \mathsf{nf} \leftarrow \mathsf{NF.Eval}(s, x), \\ \pi \leftarrow \mathsf{NF.Prove}(\mathsf{cm}_s, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

– **Uniqueness:** For each input $x$, commitment $\mathsf{cm}$, only one nullifier $\mathsf{nf}$ can be generated:

$$\text{if} \quad \mathsf{NF.Verify}(\mathsf{cm}, x, \mathsf{nf}_0, \pi_0) = \mathsf{NF.Verify}(\mathsf{cm}, x, \mathsf{nf}_1, \pi_1) = 1 \quad \text{then} \quad \mathsf{nf}_0 = \mathsf{nf}_1$$

– **Pseudorandomness:** The nullifier output is indistinguishable from random for any input not previously queried, defined by the following game $\mathcal{G}_{\mathcal{A}}^{\mathrm{nf}}$:

1. The nullifier challenger samples $s \leftarrow\$ \{0, 1\}^\lambda$, computes $\mathsf{cm} = \mathsf{CM.Com}(s)$, and sends $\mathsf{cm}$ to $\mathcal{A}$.
2. $\mathcal{A}$ submits evaluation queries $x_1, \ldots, x_Q \in \mathcal{X}$, and receives $(\mathsf{nf}_i, \pi_i)$ for each query, where $\mathsf{nf}_i = \mathsf{NF.Eval}(s, x_i)$ and $\pi_i \leftarrow \mathsf{NF.Prove}(\mathsf{cm}, x_i)$.
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 $\mathsf{nf}_0^* = \mathsf{NF.Eval}(s, x_*)$, samples $\mathsf{nf}_1^* \leftarrow\$ \mathcal{N}$ uniformly at random, then samples $b \leftarrow\$ \{0, 1\}$ and sends $\mathsf{nf}_b^*$ 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 nullifier scheme satisfies pseudorandomness if for all PPT adversaries $\mathcal{A}$:

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

## 4.3 Preliminaries

### 4.3.1 Cryptographic Assumptions

**Definition 4.2 (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* $\mathsf{negl}$ *such that:*

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

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

**Definition 4.3 ($q$-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* $\mathsf{negl}$ *such that:*

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

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

### 4.3.2 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 4.4 (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 $\Pi$ where:*

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

A secure VRF must satisfy the following properties:

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

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

- **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 $\mathcal{G}_{\mathcal{A}}^{\mathrm{vrf}}$:

  1. The VRF challenger samples $(s, pk) \leftarrow \mathsf{VRF.Gen}(1^\lambda)$, and sends $pk$ to $\mathcal{A}$.
  2. $\mathcal{A}$ submits evaluation queries $x_1, \ldots, x_Q \in \mathcal{X}$, and receives $(y_i, \pi_i)$ for each query, where $y_i = \mathsf{VRF.Eval}(s, x_i)$ and $\pi_i \leftarrow \mathsf{VRF.Prove}(s, x_i)$.
  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 $y_0^* = \mathsf{VRF.Eval}(s, x_*)$, 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 $\mathcal{A}$:

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

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

**Definition 4.5 (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 $pk = g^s$.*
- $\mathsf{VRF.Eval}(s, x)$*: Compute $y = e(g, \tilde{g})^{1/(sk+x)}$.*
- $\mathsf{VRF.Prove}(s, x)$*: Compute $\pi = \tilde{g}^{1/(sk+x)}$.*
- $\mathsf{VRF.Vfy}(pk, x, y, \pi)$*: Check $e(g^x \cdot pk, \pi) \overset{?}{=} e(g, \tilde{g})$ and $y \overset{?}{=} e(g, \pi)$.*

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

### 4.3.3 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.

### 4.3.4 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 $\mathsf{r}$, 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.

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

The Dodis-Yampolskiy [DY05] VRF generates unique, pseudorandom nullifiers $y$ and their proofs of correctness $\pi$. We start with the question of removing pairings to improve efficiency. DY consists of $(y, \pi)$:

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

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

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

Security relies on the $q$-DBDHI problem which states that given $(g, g^x, g^{x^2}, \ldots, g^{x^q}, \tilde{g})$, no $\mathsf{PPT}$ 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 $(x', y', \pi')$ pairs.

### 4.4.1 Technical Challenge of Removing Pairings from Dodisy Yampolskiy VRF

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

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

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

The insight is to reverse the verification approach. Instead of trying to derive $g^{1/(s+x)}$ from $g^{s+x}$ or cancel with a reciprocal, we use a zero knowledge proof $\Sigma$-protocol to verify that $y$ raised to the power $(s + x)$ 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:

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

### 4.4.2 New Sigma Protocol and Construction for Prime Order Dodis Yampolskiy VRF

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:

– $\mathsf{VRF.Gen}(1^\lambda) \to (s, pk)$: Sample $sk \leftarrow\$ \mathbb{Z}_p^*$, compute $pk = g^s$, and output $(s, pk)$.

– $\mathsf{VRF.Eval}(s, x) \to y$: Compute $y = g^{1/(sk+x)} \in \mathbb{G}$.

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

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

*Remark 4.6.* Verifying $\mathsf{VRF.Verify}(pk, x, \mathsf{VRF.Eval}(s, x) \to y) \to 1$ 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 $\Sigma$-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 $pk = g^s$, $y = g^{1/(sk+x)}$

**Relation:**

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

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: $g^z \overset{?}{=} T_1 \cdot (pk \cdot g^x)^c$ and $y^z \overset{?}{=} T_2 \cdot g^c$

### 4.4.3 Security Analysis

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

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

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

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 $\pi'$. 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 $(g, g^s, \ldots, g^{(sk)^q})$ for polynomial $q$. The $\Sigma$-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.

### 4.4.4 Performance Evaluation - My DY VRF Construction is 3 - 6x More Efficient

I implemented our constructions [Pol25] using the arkworks cryptography library [ark22] in Rust and evaluated performance on a MacBook Air M2 with 16GB RAM. Table 7 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[6]. Results are available in the figure 13 and table 4.4.4

**Table 7.** VRF Comparison between Original DY and Our Prime Order DY VRF Construction

| **Scheme** | **Curve** | **Eval + Prove (ms)** | **Verify (ms)** | **Total (ms)** |
|---|---|---|---|---|
| My DY Construction4.4.2 | Ed25519 | 0.21 | 0.37 | 0.58 |
| | secp256k1 | 0.25 | 0.42 | 0.67 |
| | BLS12-381 | 0.41 | 0.71 | 1.12 |
| Original DY [1] [DY05] | BLS12-381 | 1.27 | 2.27 | 3.54 |

[1]I 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.

[6] 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

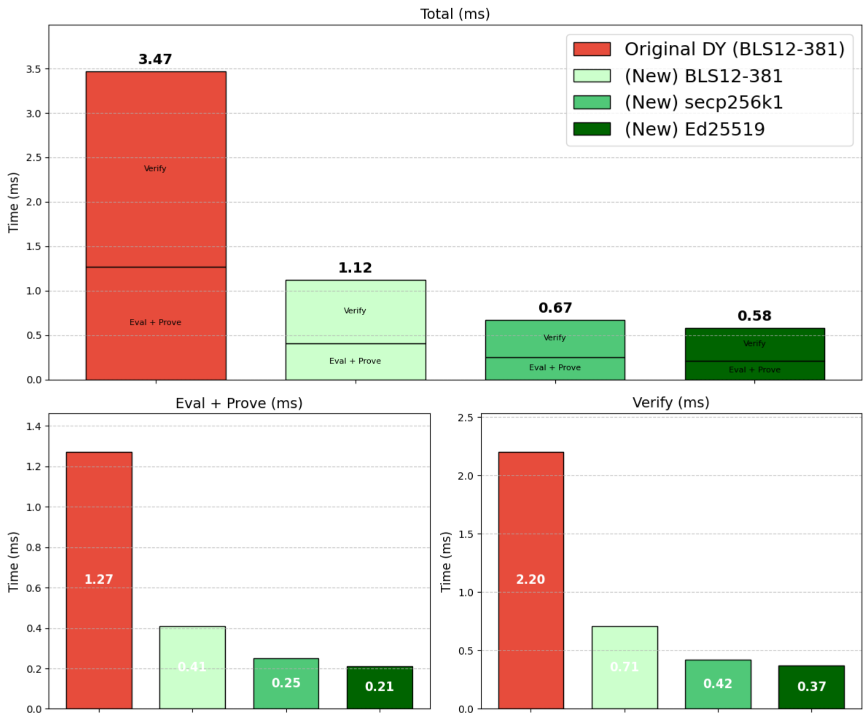

**Fig. 13.** DY VRF Performance Comparison - Our Construction is 3 - 6x more efficient

## 4.5 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 $pk = g^s$ and input $x$ during verification, which isn't sufficient for our nullifier definition. 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.

Recall the $q$-DDHI problem, given $y = g^x, g^{x^2}, \ldots, g^{x^q}$, and $q$ queries to the oracle, no adversary can distinguish $g^{1/x}$ from a random group element with non-negligible advantage. My approach starts with a $\Sigma$-protocol that proves knowledge of $1/x$ from a user with committed $x$. I then extend that to proving knowledge of $1/s + x$ and extend that to prove knowledge of $1/s + x$ within a commitment.

### 4.5.1 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.

### 4.5.2 New Zero-Knowledge Proof for Knowledge of an 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 $\mathsf{cm}$. 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 4.7.** *Protocol 4.5.2 is a Σ-protocol for* $\mathcal{R}_{\mathsf{DDHI}}$.

*Proof (Sketch).* We verify the three properties of a $\Sigma$-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.* While the q-DDHI assumption implies that a single value $g^{1/x}$ is indistinguishable from random when $x$ is hidden, this protocol reveals additional information. Specifically, when an adversary observes both $x$ (through its commitment) and $y = g^{1/x}$, the output is no longer pseudorandom. This is because there exists a verification relationship $y^x = g$ that a truly random element would not satisfy with significant probability. However, this doesn't compromise the security of our protocol, as we're proving knowledge of the relationship rather than relying on its pseudorandomness.

### 4.5.3 New Zero-Knowledge Proof of Knowledge of Inverse Linear Relation

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

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

**Construction** The protocol extends my single-commitment construction by (1) proving knowledge of openings for $\mathsf{cm}_1$ and $\mathsf{cm}_2$ simultaneously, and (2) verifying $y^{s+x} = g$ where $s + x$ combines values from distinct commitments. The consistency check $z_m = z_s + z_x$ prevents substitution attacks where an adversary might use different linear combinations across verification equations.

**Protocol 4: Proof of Knowledge of 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 $(s, x, \mathsf{r}_1, \mathsf{r}_2)$ 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:**

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

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:

$$z_s = a_{sk} + c \cdot sk \qquad z_x = a_x + c \cdot x \qquad z_m = (a_{sk} + a_x) + c \cdot (sk + x)$$
$$z_{r_1} = a_{r_1} + c \cdot r_1 \qquad z_{r_2} = a_{r_2} + c \cdot r_2$$

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

4. **Verification:** Verifier checks:

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

**Security Analysis**

**Theorem 4.8.** *Protocol 4.5.3 is a $\Sigma$-protocol for $\mathcal{R}_{\mathsf{nf}}$.*

*Proof (Sketch).*

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

### 4.5.4 Constructing a Deterministic Nullifier from PoK Inverse Linear Relation

My 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 $\pi \in \Pi \ = \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p$. 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

- $\mathsf{dVRF.Eval}(s, x) \to y$: $y = g_3^{1/(s+x)}$
- $\mathsf{dVRF.Prove}(\mathsf{cm}_1, \mathsf{cm}_2, s, x, y) \to \pi$: Run protocol 4.5.3, output $\pi, \mathsf{cm}_y$
- $\mathsf{dVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, y, \pi) \to \{0, 1\}$: Output 1 if $\pi$ verifies $\mathsf{cm}_1, \mathsf{cm}_2, y$ correctly per the $\Sigma$-protocol 4.5.3, else 0.

#### Security Properties

**Theorem 4.9 (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 $\mathsf{cm}_2 = g_2^x g^{r_2}$:*

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

*Proof.* Correctness follows directly from the completeness of Protocol 4.5.3. Since honest execution ensures all verification equations in the underlying Sigma protocol hold, the VRF verification checks pass.

**Theorem 4.10 (Uniqueness).** *Under the binding property of Pedersen commitments and the discrete logarithm assumption, for any $\mathsf{cm}_1$ and $\mathsf{cm}_2$, 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 $\pi$.*

*Proof (Sketch).* The binding property of Pedersen commitments ensures that $\mathsf{cm}_1$ commits to $sk$ and $\mathsf{cm}_2$ 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 $y^{s+x} = g$ and $y'^{s+x} = g$, implying $(y'/y)^{s+x} = 1$. 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 4.11 (Pseudorandomness).** *Under the q-DDHI assumption, the deterministic nullifier VRF satisfies pseudorandomness. For any PPT adversary $\mathcal{A}$:*

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

*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.

### 4.5.5 My 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 $(s, x)$ 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 $(s, x)$, 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.

My solution commits to the nullifier as:

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

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 Knowledge 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:** $(s, x, r_1, r_2, r_3, r_4) \in \mathbb{Z}_p^6$

**Relation:**

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

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:

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

4. **Verifier** accepts iff:

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

**Security Analysis** We establish the security of my 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 $(s, x, r_1, r_2, r_3, r_4)$ 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:

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

The verifier checks:

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

$$1) \qquad T_1 \cdot \mathsf{cm}_1^c = g_1^{a_{sk}} g^{a_{r_1}} \cdot (g_1^{sk} g^{r_1})^c = g_1^{a_{sk}+c\cdot sk} g^{a_{r_1}+c\cdot r_1} = g_1^{z_s} g^{z_{r_1}}$$

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

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

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

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

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

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

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. $z_m \overset{?}{=} z_s + z_x$

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

All equations hold, confirming completeness.

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

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

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 $\mathsf{cm}_3$ commits to $g_3^{1/(sk+x)} g^{r_3}$. This confirms extractability.

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

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

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

### 4.5.6 New Rerandomizable Nullifier from PoK Committed Inverse Linear Relation

**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 $\mathsf{cm}_4$ for verification.

- $\mathsf{pVRF.Eval}(s, x) \to \mathsf{cm}_y$: Sample $r_3 \leftarrow\$ \mathbb{Z}_p^*$, compute $\mathsf{cm}_y = g_3^{1/(s+x)} g^{r_3}$, sample $r_4 \leftarrow\$ \mathbb{Z}_p^*$ compute $\mathsf{cm}_4 = g_3 g^{r_3 \cdot (s+x)+r_4}$ return $\mathsf{cm}_y, \mathsf{cm}_4$
- $\mathsf{pVRF.Prove}(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4, s, x, r_1, \ldots, r_4) \to \pi$: Run protocol 4.5.3, output $\pi$
- $\mathsf{pVRF.Verify}(\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4, \pi) \to \{0, 1\}$: Output 1 if $\pi$ verifies $\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_y, \mathsf{cm}_4$ correctly per the $\Sigma$-protocol 4.5.5, else 0.

**Security Properties**

**Theorem 4.12 (Correctness).** *The probabilistic nullifier VRF is correct following from the completeness of Protocol 4.5.5.*

**Theorem 4.13 (Uniqueness).** *For fixed* $\mathsf{pk}$ *and* $\mathsf{cm}_2$*, 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 4.14 (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* $\mathsf{Exp}^{\mathsf{unlink-b}}$ *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 4.5.5, 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 $(s, x)$ pair.

## 4.6 Performance Evaluation

I implemented my constructions [Pol25] using the arkworks cryptography library [ark22] in Rust and evaluated performance on a MacBook Air M2 with 16GB RAM. Table 4.6 compares my constructions on BLS12-381 curve against baseline approaches, focusing on the critical operations of evaluation, proof generation, and verification. Table 8 shows our constructions built on different, non-pairing curves.

My evaluation reveals several key findings:

1. **Pairing Elimination:** My 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 my 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

**Table 8.** Performance of my VRF Constructions Across Elliptic Curves (time in ms)

| Curve | Scheme | Eval + Prove | | Verify | |
|---|---|---|---|---|---|
| | | ms | Speedup | ms | Speedup |
| BLS12-381 | Prime Order DY 4.4.2 | 0.41 | - | 0.71 | - |
| | Prime Order, Anon. Deterministic 4.5.3 | 1.15 | - | 1.08 | - |
| | Prime Order, Anon. Probabilistic 4.5.5 | 2.01 | - | 1.65 | - |
| Ed25519 | Prime Order DY 4.4.2 | 0.21 | 2.0× | 0.37 | 1.9× |
| | Prime Order, Anon. Deterministic 4.5.3 | 0.56 | 2.0× | 0.56 | 1.9× |
| | Prime Order, Anon. Probabilistic 4.5.5 | 1.04 | 1.9× | 0.84 | 2.0× |
| secp256k1 | Prime Order DY 4.4.2 | 0.25 | 1.6× | 0.42 | 1.7× |
| | Prime Order, Anon. Deterministic 4.5.3 | 0.66 | 1.8× | 0.66 | 1.6× |
| | Prime Order, Anon. Probabilistic 4.5.5 | 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.

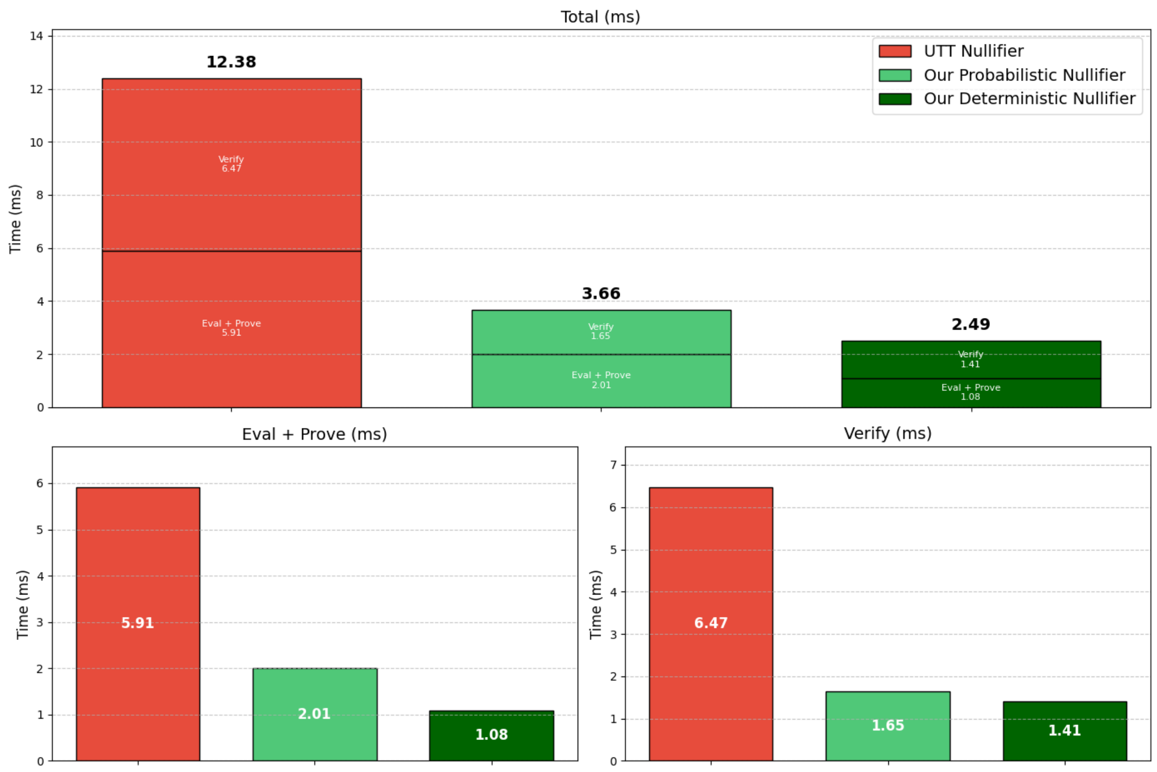

**Fig. 14.** Nullifier Performance Comparison - Our Constructions are 3x - 5x more efficient

### 4.6.1 Applications and Integration of Nullifiers

In this section, we detail the application of nullifiers in privacy-preserving systems and their integration into the credential wallet. A nullifier, built using a pairing-free Verifiable Random Function (VRF) and Sigma-protocols, ensures efficient uniqueness and revocability with strong privacy guarantees.

**Applications Sybil Resistance.** The deterministic nullifier $y = g^{1/(sk+x)}$, where $sk$ is the user's secret key and $x$ is a context identifier, enforces uniqueness per user-context pair. For instance, in anonymous voting, a user computes $y = g^{1/(sk+\text{"Election2024"})}$ and submits it alongside a zero-knowledge (ZK) proof of correctness, without disclosing $sk$ or $x$. The system verifies the proof and checks $y$ against a public list of used nullifiers, rejecting duplicates to prevent double-voting. This leverages Sigma-protocols in prime-order groups for efficiency, avoiding pairing operations.

**Revocation.** The committed nullifier $cm_y = g_3^{1/(sk+x)} g^{r_3}$, where $r_3$ is randomness, enables privacy-preserving revocation. In a service access scenario, a user registers with $y = g^{1/(sk+\text{"ServiceA"})}$. Upon revocation, $y$ is published to a public list. For access, the user presents $cm_y$ with a ZK proof that $cm_y$ commits to a nullifier absent from the list, using set non-membership techniques. The randomness $r_3$ ensures unlinkability across presentations.

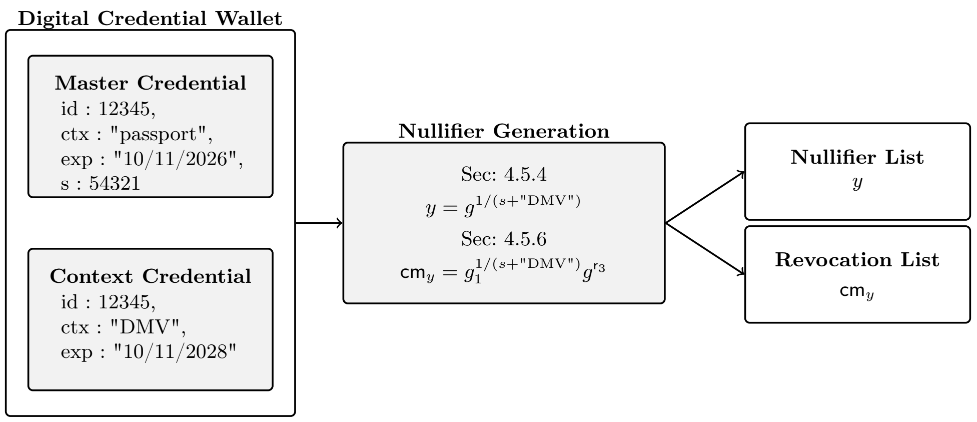

**Fig. 15.** A nullifier scheme for privacy-preserving protocols, illustrated with a credential hierarchy binding context credentials (e.g., driver's license) to a master credential (e.g., passport). The deterministic nullifier prevents sybil attacks, while the probabilistic nullifier supports revocation, with applications extending to voting, e-cash, and pseudonymous identities.

**Integration into the Credential Wallet** The credential wallet employs nullifiers for secure credential management:

- **Storing and Presenting Credentials.** For a credential tied to context $x$, the wallet computes $y = g^{1/(sk+x)}$. During presentation, it provides $y$ and a ZK proof of correctness, validating the credential without linking presentations.
- **Revocation Handling.** The wallet either checks $y$ against a revocation list or generates a ZK proof demonstrating non-revocation, preserving user privacy.

The nullifier's deterministic design guarantees uniqueness, while its efficient verification in prime-order groups suits real-time use. This integration ensures the credential wallet balances security, privacy, and practicality.

## 4.7 Summary and Future Work

This chapter first formally introduced nullifiers. I then enhanced the Dodis Yampolskiy VRF by restricting it to a single prime-order group, removing pairings, and through a new $\Sigma$-protocol, I achieve a 3-5x efficient improvement demonstrated in Figure 13. Secondly, I created two nullifier schemes, en route, I created a $\Sigma$-protocol, zero-knowledge proof of knowledge of an inverse exponent. I then extended the inverse exponent protocol to handle linear combinations of committed values which becomes the proof in the Deterministic Nullifier Construction. Furthermore, I created a separate approach if a committed nullifier is required for the output, I call this a Rerandomizable Nullifier as it's committed and can be rerandomzied with $r$ each invocation.

### 4.7.1 Future Work

I have 4 different directions in mind for future work

1. **Formalize Nullifier Security Definition:** As the primitive is found in different protocols, good care should be put into developing a flexible construction-agnostic model and security properties, similarly to the Anonymous Credential model by [FHS19].

2. **Performance Benchmarking:** As ECDSA signatures are foundational to blockchain systems, comprehensive benchmarking of our nullifier scheme against SNARK-based approaches similar to [GG22] would provide valuable performance insights for practical implementations.

3. **Alternative Assumptions:** Investigating nullifiers derived from structure-preserving signature schemes with embedded group elements (particularly SPS-EQ) could yield constructions with additional properties like stronger unlinkability or more efficient or flexible verification.

4. **Post-Quantum Security:** Developing post-quantum secure nullifier schemes. This could involve adapting our techniques to post-quantum primitives or finding novel ways to transform classical signatures into quantum-resistant nullifiers while preserving their privacy properties.

# Chapter 5

# Threshold-Issued, Sybil-Resistant Private Identity System from Anonymous Credentials and Nullifiers

## 5.1 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 4), adding just 2.49ms to issuance overhead to credential issuance (Section 5.5.2) 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 3 and Credential Relationship Binding Nullifier (CRBN) from Chapter 4. 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. My implementation can be found on GitHub[7]

### Chapter Roadmap

The remainder of this chapter is structured as follows: Section 5.2 covers cryptographic preliminaries. Section 5.3 presents the T-SIRIS system model and security requirements. Section 5.4 details our construction and protocols. Section 5.5 evaluates our performance against state-of-the-art alternatives.

### 5.1.1 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.

[7] https://github.com/sampolgar/t-siris

**CanDID** CanDID deduplicates a user attribute, e.g. social security number using MPC, storing a unique value in a table $T_{\mathrm{Dedup}}$. 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 $y = g^{1/(s+\mathrm{ctx})} \in \mathbb{G}_1$ with user key $k \in \mathbb{Z}_p$ and context ctx. This is anonymous and many times faster than tACT [RAR+24] (see Section 5.5).

– **Efficient Deduplication and Verification:** T-SIRIS achieves fast deduplication using CRBN, computing a nullifier $y = g^{1/(s+\mathrm{ctx})} \in \mathbb{G}_1$ 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 $5\times$ faster than S3ID's tACT [RAR+24]. Verification scales sublinearly with attributes $n$, achieving up to $44.1\times$ speedup over tACT for $n = 64$ (see Section 5.5).

– **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 $\Pi^{\mathcal{R}_{\mathrm{null}}} = \mathrm{ZKPoK}\{(k) : y = g^{1/(s+\mathrm{ctx})}\}$ 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 $44.1\times$ faster verification for $n = 64$ attributes (see Section 5.5).

– **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 $N$.

### 5.1.2 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 3 and the Credential Relationship Binding Nullifier (CRBN) from Chapter 4, 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 $y = g^{1/(s+\mathrm{ctx})} \in \mathbb{G}_1$ 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 $44.1\times$ 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 3).

4. **Non-transferability and Identity Binding:** T-SIRIS embeds a user identifier $\mathsf{id}$ 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 9.** 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 i.e. verify against predicates from multiple credentials
[‡] Credentials bound to the true owner (Section 5.4).
[§] Support for complex predicates, e.g., range proofs (Section 5.4.4).
[¶] Anonymity against malicious issuers (Section 2.3.2).
[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.

## 5.2 Preliminaries

**Definition 5.1 (Shamir Secret Sharing).** *A $(t, n)$ secret sharing scheme* $\mathsf{SS}$ *consists of the following* $\mathsf{PPT}$ *algorithms over message space $\mathcal{X}$:*

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

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

### 5.2.1 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 $\mathsf{sk}_i$ and a share verification key $\mathsf{vk}_i$. This enables subsets of at least $t+1$ signers to collaboratively produce a signature that verifies under the combined verification key $\mathsf{vk}$, 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 4, 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 5.2 (PS Threshold Signatures over Pedersen Commitments).** *A PS threshold signature scheme* $\mathsf{RS}$ *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 $\lambda$, corruption threshold $t$, number of parties $n$, and attribute count $\ell$. Outputs commitment key* $\mathsf{ck}$*, verification key* $\mathsf{vk}$*, 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* $\mathsf{sk}_i$*, commitments* $\mathsf{cm}_k$ *with ZK proofs $\pi_k^{\mathsf{zkpok}}$, 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* $\mathsf{vk}_i$*, 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 $\sigma$.*

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

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

**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 $\mathsf{cm}$ 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, $\ell$ is the credential message length. Outputs $\mathsf{ck}, \mathsf{vk}$ 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 $\psi_k$ 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 $\mathsf{ck}$ 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 $\mathsf{ck}$ and verifies $\{\mathsf{ZK.Verify}(\Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k))\}_{k\in[\ell]}$
- Signer parses $\mathsf{sk}_i$ 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 $\mathsf{vk}_i$ 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 $\mathsf{ck}$ 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 $\sigma$ as $(\sigma_1, \sigma_2)$ 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 $\sigma$ as $(\sigma_1, \sigma_2)$, The prover $\mathcal{P}$ runs a Proof of Knowledge protocol with the following relation

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

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

## 5.3 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 4) 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 3) to issue credentials with $t$ of $n$ issuers, ensuring robustness.
- **Deduplication**: Employs nullifiers from Chapter 4 for efficient sybil resistance without costly operations.
- **Expressive Proofs**: Leverages Chapter 2'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, or 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 $\lambda$ and attribute vector length $\ell$. We use $\mathsf{RS.Setup}, \mathsf{RS.Rand}, \mathsf{RS.Ver}$ from the rerandomizable signature scheme.

**Definition 5.3 (tSIRIS).**

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

**Definition 5.4 (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:*
   - $\mathsf{Verify}(\mathsf{cred}', \phi, \pi) = 1$
   - $\mathsf{cred}'$ *was not legitimately issued to a corrupted user*
   - $\phi(\vec{m}) = 0$*, where $\vec{m}$ are the attributes in* $\mathsf{cred}'$

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

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

### 5.3.1 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 5.5 (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 $\mathsf{cred}_{master}$, context $\mathsf{ctx}$ and two context credentials $(\mathsf{cred}_1, \mathsf{cred}_2)$*
8. *The experiment returns 1 if:*
   - $\mathsf{Verify}(\mathsf{cred}_{master}, \phi_{master}, \pi_{master}) = 1$
   - $\mathsf{Verify}(\mathsf{cred}_1, \phi_{\mathsf{ctx}}, \pi_1) = \mathsf{Verify}(\mathsf{cred}_2, \phi_{\mathsf{ctx}}, \pi_2) = 1$
   - $\mathsf{cred}_1$ *and* $\mathsf{cred}_2$ *have the same context* $\mathsf{ctx}$
   - $\mathsf{cred}_1$ *and* $\mathsf{cred}_2$ *were issued using the same* $\mathsf{cred}_{master}$

– $\mathsf{cred}_1 \neq \mathsf{cred}_2$ *(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* $\mathsf{negl}$ *such that:*

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

### 5.3.2 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 5.6 (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 $\mathsf{cred}^i_{master}$ with attributes $\mathsf{attrs}_i$*
   - *User $i$ obtains a context credential $\mathsf{cred}^i_{\mathsf{ctx}}$ for context $\mathsf{ctx}$*
6. *$\mathcal{A}$ outputs predicate $\phi$ 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* $\mathsf{negl}$ *such that:*

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

### 5.3.3 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.

## 5.4 Our Construction

### 5.4.1 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 $\mathsf{s}$, while context credentials are bound to this master credential through nullifiers derived from $\mathsf{s}$ and a context $\mathsf{ctx}$. 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 $\Sigma$-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 $\mathsf{cred}$ 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 5.4.2, 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 $\mathsf{cred_m} = (\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{vk_m})$, including a rerandomizable, threshold-signature over commitment.

### 5.4.2 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 $s_1$, $\mathsf{cm}_1 = \mathsf{CM.Com}([s_1]; r)$[8] 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 $\mathsf{cm}_2, s_2$ 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 $\mathsf{DKG.Combine}(\{\mathsf{s}_{2_\mathsf{i}}\}_{\mathsf{i} \in \mathsf{S}})$, 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.

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

[8] $\mathsf{cm}$ contains more than just the VRF key $\mathsf{s}$, we simplify in this explanation for brevity

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., $\mathsf{cm_m} = \mathsf{CM.Com}([id, \ldots]; r_1), \mathsf{cm_c} = \mathsf{CM.Com}([id, \ldots]; r_2)$ 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 $\phi$ 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 $\mathsf{cm_c} = \mathsf{CM.Com}([id, \mathsf{ctx_c}, \ldots]; r_c)$ and proves their $id$ is consistent between the two commitments. The user proves this relation:

$$\mathcal{R}_\phi : \left\{ \begin{array}{l|l} (\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{cm_c}), & \mathsf{RS.Ver}(\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{vk_m}) = 1 \wedge \\ (id, s, \mathsf{ctx_m}, \mathsf{ctx_c}, r_m, r_c) & \mathsf{cm_m} = \mathsf{CM.Com}([id, \mathsf{ctx_m}, s, \ldots]; r_m) \ \wedge \mathsf{ctx_m} = \texttt{"master"} \wedge \\ & \mathsf{cm_c} = \mathsf{CM.Com}([id, \mathsf{ctx_c}, \ldots]; r_c) \ \wedge \mathsf{ctx_c} = \texttt{"DMV"} \end{array} \right\}$$

The Context Credential issuer verifies the correctness of $\Pi_\phi$ and runs their signing protocol over $\mathsf{cm_c}$.

### 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 4.5.4 as $\mathsf{nullifier} = g^{1/s+ctx} \leftarrow \mathsf{dVRF.Eval}(s, ctx)$. The nullifier's proof of correctness $\pi \leftarrow \mathsf{dVRF.Prove}(\mathsf{cm_m}, \mathsf{cm_c}, s, ctx, \mathsf{nullifier})$, outlined here: 4.5.3, ensures it was generated by $s$ in the master credential and $\mathsf{ctx_c}$ from the Context Credential. To prevent replay attacks, $\mathsf{nullifier}$ is an input into the context credential issuance process $\pi$ is combined with the previous $\Sigma$-protocol.

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

### Revocation

A popular efficient accountable-privacy pattern for revocation is seen in anonymous payments [DGK+20] and private identity systems [WGW+23]: 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.

### 5.4.3 Protocol Details

**OrgKeyGen** $\mathsf{OrgKeyGen}(1^\lambda, \ell, t, n) \rightarrow (\mathsf{BG}, \mathsf{ck}, \mathsf{vk}, \{\mathsf{sk}_i, \mathsf{vk}_i\}_{i \in [\ell]}, \{[s_2]_i\}_m$ 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 $\{\{s_2\}_i, \mathsf{vk}_{s_2}\}_m = \{\{s_2\}_i, \mathsf{vk}_{s_2}\}_1, \dots, \{\{s_2\}_i, \mathsf{vk}_{s_2}\}_m$ 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 $\mathsf{vk}_{s_2}$, 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 $\mathsf{tPrepare}(\mathsf{ck}, h, \mathsf{attrs}) \rightarrow (\{\mathsf{cm}_k, \Pi^{\mathcal{R}_{\mathsf{com}}}(\mathsf{cm}_k, m_k, r_k)\}_{k\in[\ell]}, h)$, 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 $\mathsf{cm}_k$ for $\mathsf{Issue}$

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

1. $|S|$ issuers run $\mathsf{tShareSign}(\mathsf{ck}, \mathsf{sk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, h) \rightarrow [\sigma^*]_i$ :
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. $\mathsf{tAggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, \{r_k\}_{k\in[\ell]}) \rightarrow \sigma$ :

**(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:** $\mathsf{ObtainContext}(\mathsf{attrs}, \mathsf{cred}_{\mathsf{m}}, \phi)$

1. User samples $\Delta_r, \Delta_u \leftarrow\$ \mathbb{Z}_p$, rerandomizes $\mathsf{RS.Rand}(\sigma_{\mathsf{m}}, \mathsf{cm}_{\mathsf{m}}) \rightarrow \sigma'_{\mathsf{m}}, \mathsf{cm}_{\mathsf{m}}'$
2. User computes $\mathsf{nullifier}, \Pi^{\mathsf{R}_{\mathsf{credc}}}$, $\mathsf{nullifier} \leftarrow \mathsf{dVRF.Eval}(s, \mathsf{ctx})$ and $\Pi^{\mathsf{R}_{\mathsf{credc}}} \leftarrow \mathsf{dVRF.Prove}(\mathsf{cm}_{\mathsf{m}}, \mathsf{cm}_{\mathsf{c}}, s, \mathsf{ctx}, \mathsf{nullifier})$, 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 $\mathsf{cm}_k$ for $\mathsf{Issue}$

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

1. $|S|$ issuers run
   (a) $\mathsf{ZK.Verify}(\mathsf{cm}_{\mathsf{m}}, \mathsf{cm}_{\mathsf{c}}, \mathsf{nullifier}, \Pi^{\mathsf{R}_{\mathsf{credc}}})$ and $\mathsf{ZK.Verify}(\{\mathsf{cm}_k, \Pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]})$[9]. The $\mathsf{cm}_k$ committing to $\mathsf{ctx}$ opens to the same index of $\mathsf{cm}_{\mathsf{c}}$ which the issuers will verify.

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

(b) $\mathsf{tShareSign}(\mathsf{ck}, \mathsf{sk}_i, \{\mathsf{cm}_k, \pi_k^{\mathsf{zkpok}}\}_{k\in[\ell]}, h) \to [\sigma^*]_i$ :

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. $\mathsf{tAggregate}(\mathsf{ck}, ([\sigma^*]_i)_{i\in S}, \{r_k\}_{k\in[\ell]}) \to \sigma$ :

**(Show, Verify)** Show and Verify follow from the non-threshold ABC system 2.5.4. The user runs $\mathsf{RS.Rand}(\sigma, \mathsf{cm}) \to \sigma', \mathsf{cm}'$ 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.

### 5.4.4 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 5.7 (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 5.2.1 inherits the EUF-CMA security of the standard PS signature scheme from Chapter 2. 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 3), 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.

#### Sybil Resistance

**Theorem 5.8 (Sybil Resistance).** *If the deterministic nullifier scheme from Chapter 4 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 4.5.4.

For each user-context pair $(s, \mathsf{ctx})$, where $s$ is the master credential key and $\mathsf{ctx}$ 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 5.9 (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 3), 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 4) 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 5.10 (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.

## 5.5 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.

### 5.5.1 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 16

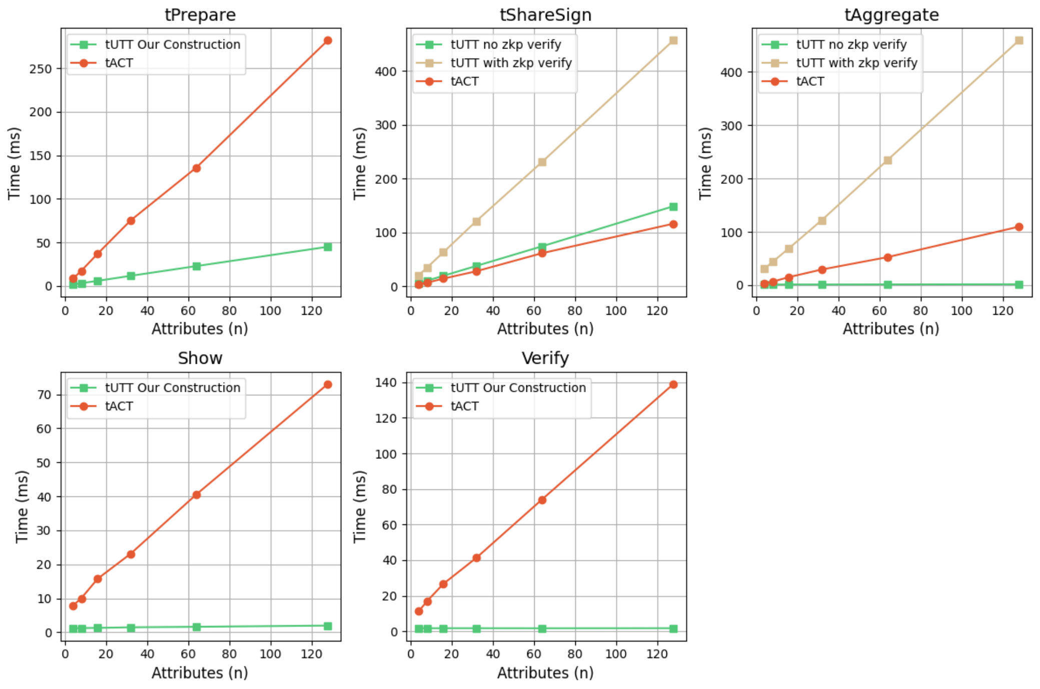

**Fig. 16.** Performance comparison when scaling by attribute count ($n$) with fixed $N = 16, t = 9$

- **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 16 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:

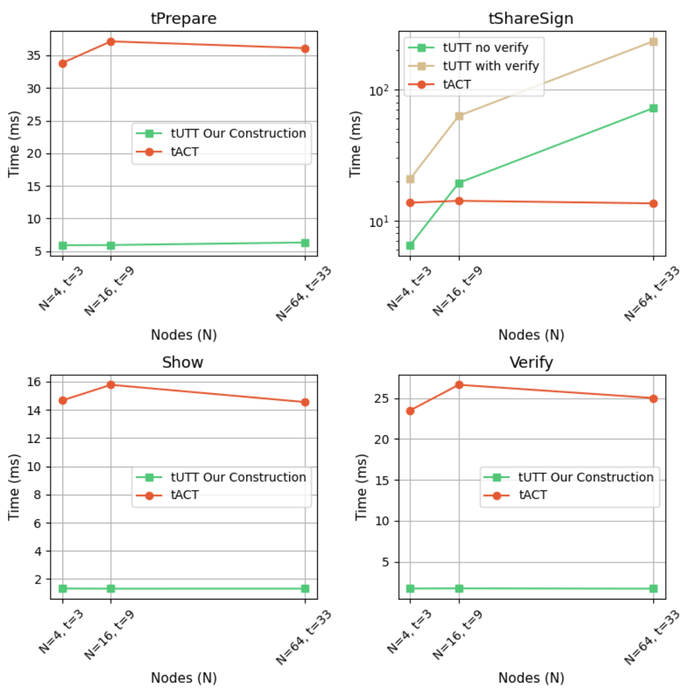

**Fig. 17.** Performance comparison when scaling by threshold size ($N$) with fixed $n = 16$

- **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)$^\dagger$ | 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. 18.** tUTT (Ours) Performance scaling with attribute count ($n$) for fixed $N = 16, t = 9$ (ms) against tACT Construction

| **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)$^\dagger$ | 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. 19.** tUTT (Ours) Performance scaling with threshold size ($N$) for fixed $n = 16$ (ms) against tACT Construction

Figure 17 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.

### 5.5.2 Threshold Identity System Evaluation

We now analyze the complete T-SIRIS system's performance. Figure 20 illustrates how each system operation scales with attribute count across different threshold configurations.

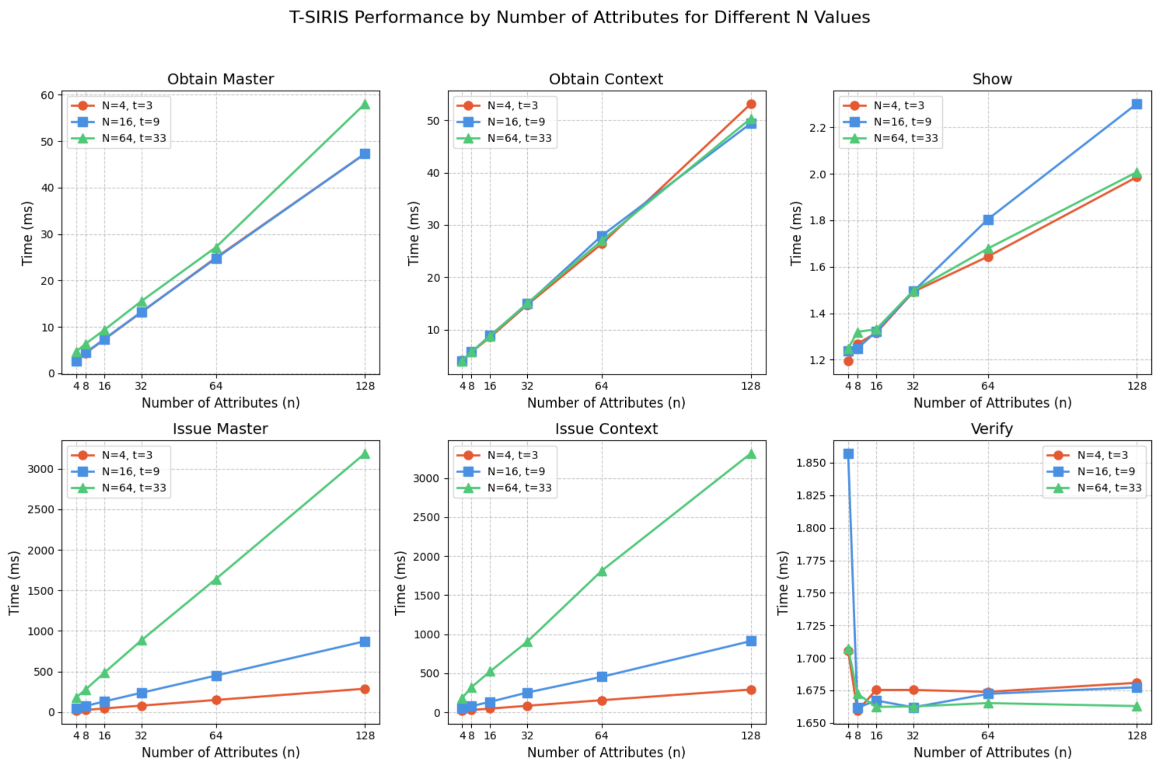

**Fig. 20.** T-SIRIS performance scaling with attribute count across operations

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 60$ms$ 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.

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.

### 5.5.3 Threshold Identity System Performance Comparison with S3ID

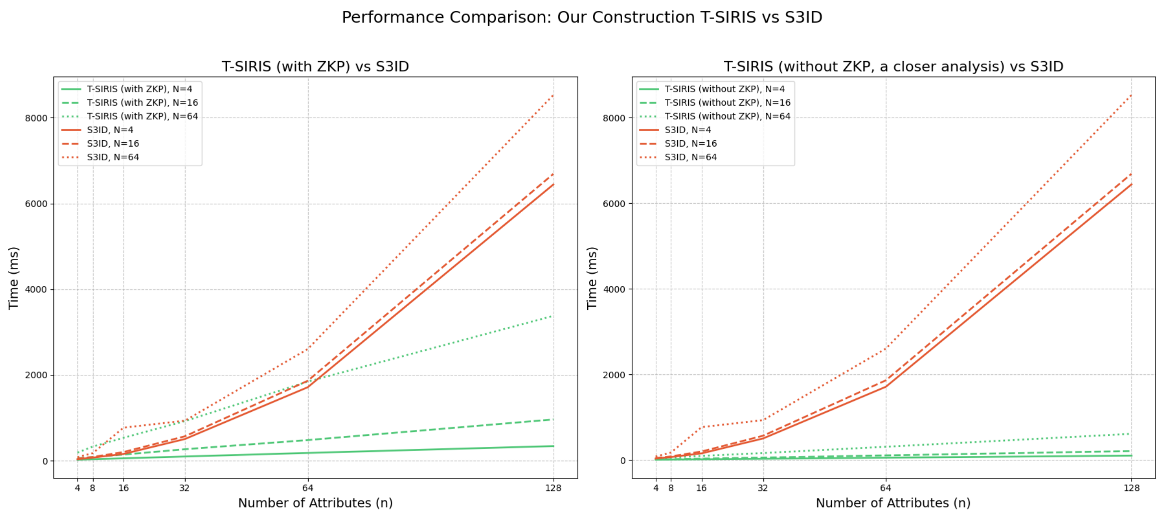

**Fig. 21.** End-to-end performance comparison between T-SIRIS and S3ID

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: 21.

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 6

# Open Source Anonymous Credentials Benchmark Library

Anonymous Credentials lack a standardized, practical evaluation tool which will hinder adoption progress. I introduce an open-source library in Rust that benchmarks state-of-the-art Attribute-Based Anonymous Credential (ABC) schemes under consistent conditions. The field suffers from inconsistent comparisons—varying languages (e.g., Python, C++, Rust) and libraries (e.g., Arkworks, Ethereum PyEcc)—making it hard to assess schemes like BBS+ and PS fairly. Our work bridges this gap, offering practitioners a tool to understand functional differences and performance trade-offs while revealing how cryptographic optimizations enhance efficiency beyond theoretical predictions. The library is available on my GitHub[10]

## 6.1 Core Contributions

My primary contribution is an open-source Rust library that implements and benchmarks leading ABC schemes (e.g., BBS+, PS, and their variants), providing a standardized framework for fair comparisons. Unlike prior evaluations, which lack consistency, our library ensures identical conditions—same hardware, security parameters, and libraries—highlighting the most efficient scheme for the critical "Show + Verify" operation (e.g., PS-UTT22 at 5.38 ms for 30 attributes, a 6.09x speedup over PS16). This tool also serves as an educational resource, helping practitioners explore constructions and select schemes for use cases like decentralized identity or Sybil-resistant voting.

My second contribution demonstrates the gap between theoretical and practical cryptography through the empirical evaluation of two optimizations:

1. **Schnorr Proofs with Multi-Scalar Multiplication (MSM):** I show that Schnorr proofs, often criticized as linear in attribute size, scale sublinearly in practice using MSM, challenging theoretical assumptions and boosting efficiency in New Generation Anonymous Credential schemes using $\mathbb{G}_1$ Schnorr proofs.

2. **Pairing Optimizations Pairings:** By leveraging Miller-Loop optimizations, I show pairing computation cost can be reduced, improving pairing-based credential performance.

These findings, validated in our library, extend beyond ABCs, offering insights for broader cryptographic applications.

## 6.2 Fair Anonymous Credential Microbenchmarks

The results table 6.4 demonstrates the standardized benchmark results from my Anonymous Credential library [Pol25], demonstrating 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 ($\mathbb{G}_1$)[11]. For 30 attributes, older schemes require computing over $3\ell$, and $2\ell$ $\mathbb{G}_T$ 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

[10] https://github.com/sampolgar/anonymous-credentials

[11] [CL04] is included in previous generation

rely on standard security assumptions, enhancing the practicality of anonymous credentials for digital identity systems.

**Table 10.** 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 |

## 6.3 Evaluating Cryptography Library Optimizations

A key challenge when assessing Anonymous Credential constructions is the use of optimizations lowering computational cost and therefore making practical benchmarks unclear and potentially unfair comparisons between different libraries. Optimizations like Multi-Scalar Multiplication and Pairing techniques lower computational cost. Below, I explore the impact of these optimizations:

### 6.3.1 Schnorr Proofs with Multi-Scalar Multiplication (MSM) are Sublinear

$\Sigma$-protocols, such as Schnorr proofs, are often characterized as having linear complexity with respect to the number of attributes. However, by leveraging multi-scalar multiplication (MSM)—an optimized algorithm widely available in cryptographic libraries—we demonstrate that these proofs achieve sublinear scaling in practice. MSM efficiently computes sums of scalar multiplications in elliptic curve groups, reducing the computational cost of proof generation and verification compared to naive approaches. This positively impacts the performance of Anonymous Credential schemes that leverage $\Sigma$-protocols. Figure 22 provides benchmark results, illustrating the sublinear scaling as the number of attributes increases.

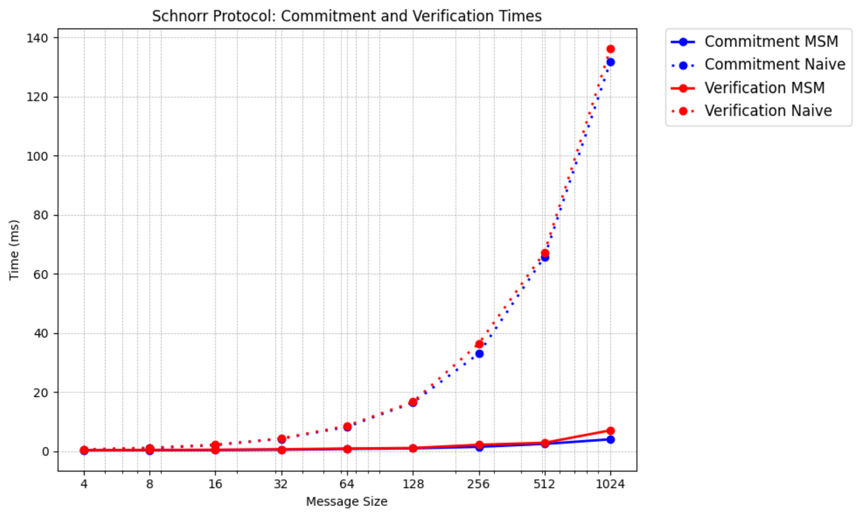

**Fig. 22.** Schnorr Protocol - Practical Benchmarks with Multi-Scalar Multiplication

### 6.3.2 Optimizations for Pairing Protocols

Pairing-based cryptography underpins many anonymous credential schemes, enabling functionalities like zero-knowledge proofs and attribute-based signatures through bilinear maps. However, pairings are computationally expensive, often dominating the runtime of schemes such as BBS+06 [ASM06] and PS16 [PS16]. Most theoretical analyses, such as those in [PS16], measure complexity as a count of pairing operations (e.g., $3\ell$ or $2\ell$ pairings for $\ell$ attributes), overlooking practical optimizations available in modern cryptographic libraries. In this subsection, we detail two key optimizations—Miller Loop with Final Exponentiation and Batch Pairing Inversion—implemented in our open-source Rust library, and demonstrate their impact on bridging the gap between theoretical complexity and practical performance.

**Miller Loop and Final Exponentiation** A pairing operation, such as the Tate or Ate pairing, denoted $e(P, Q)$, comprises two phases: the Miller Loop, which computes a rational function over elliptic curve points, and the Final Exponentiation, which maps the result to a unique element in the target group $\mathbb{G}_T$. For a product of $\ell$ pairings:

$$\prod_{i=1}^{\ell} e(g_i, \hat{h}_i),$$

a naive implementation computes $\ell$ full pairings, each requiring a Miller Loop and a Final Exponentiation, followed by $(\ell - 1)$ multiplications in $\mathbb{G}_T$. This incurs a cost of approximately $\ell P + (\ell - 1)M_{\mathbb{G}_T}$, where $P$ is the cost of one pairing and $M_{\mathbb{G}_T}$ is a multiplication in $\mathbb{G}_T$. However, by exploiting bilinearity, we optimize this as:

$$\prod_{i=1}^{\ell} e(g_i, \hat{h}_i) = \mathrm{FinalExp}\left(\prod_{i=1}^{\ell} \mathrm{MillerLoop}(g_i, \hat{h}_i)\right).$$

Here, the computation reduces to $\ell$ Miller Loops (each roughly 40% of a pairing's cost, or $0.4\ell P$) and one Final Exponentiation (approximately 60% of a pairing, or $0.6P$), eliminating the need for additional $\mathbb{G}_T$ multiplications. For large $\ell$, this yields significant savings, as the single Final Exponentiation amortizes across all pairings.

**Batch Pairing Inversion** Verification of pairing equalities, such as $e(a, b) = e(c, d)$, is common in credential schemes. Naively, this requires two full pairing computations and a comparison in $\mathbb{G}_T$. Batch Pairing Inversion transforms this into a single check:

$$e(a, b) \cdot e(c^{-1}, d) \overset{?}{=} 1_{\mathbb{G}_T},$$

where $c^{-1}$ is the inverse of $c$ in its source group. By combining the pairings into a product, this reduces the cost from two standalone pairings (approximately $2P$) to one optimized pairing (approximately $P$), cutting computation time by nearly 50% in multi-equality scenarios, such as verifying credentials with multiple attributes.

**Practical Impact** These optimizations, implemented in our Rust library using frameworks like Arkworks, significantly enhance the performance of pairing-based anonymous credentials. For instance, older schemes like PS16 and BBS+06, which rely heavily on $\mathbb{G}_T$ operations, see verification times of 32.78 ms and 39.21 ms for 30 attributes (Table 10). By applying Miller Loop and Final Exponentiation optimizations, we reduce the pairing overhead, while Batch Pairing Inversion streamlines equality checks in verification. Benchmarks, visualized in Figure 23, show that these techniques yield speedups of up to 2x for pairing-intensive operations compared to unoptimized implementations. When combined with the shift to $\mathbb{G}_1$-based proofs in newer schemes like PS-UTT22 and BBS+2016, the cumulative effect is a 6x performance boost, underscoring the critical role of practical optimizations.

## 6.4 Impact and Conclusion

My work addresses the challenge of inconsistent ABC evaluations by delivering a standardized Rust-based benchmarking library. Through rigorous empirical analysis, I uncover performance improvements as sent throughout the Thesis. These results bridge the divide between theoretical cryptography

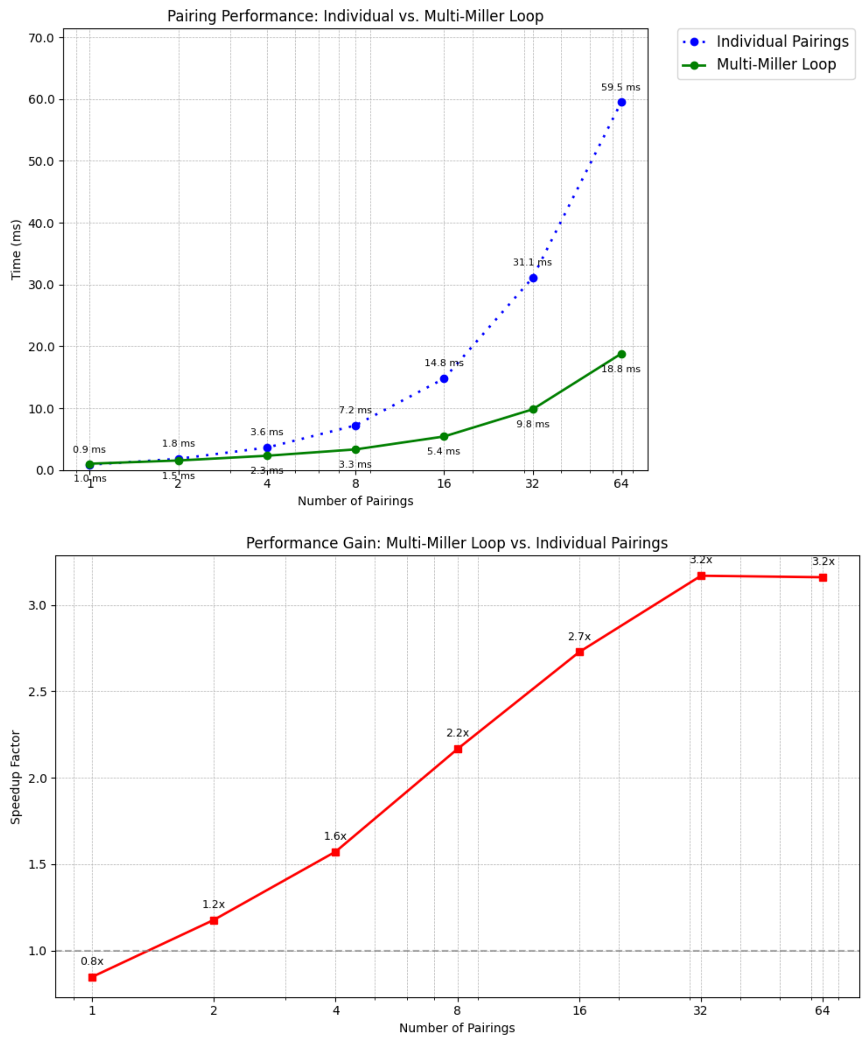

**Fig. 23.** Elliptic Curve Pairings - Practical Benchmarks with Miller-Loop Intermediate Computation

and real-world implementation, facilitating the adoption of ABCs in applications such as decentralized identity and secure voting systems. While currently focused on Rust-compatible schemes, future enhancements could integrate additional cryptographic primitives and optimizations, broadening its scope. Ultimately, this library empowers the cryptographic community with a resource for learning and implementing privacy-preserving technologies.

**Table 11.** Performance of Anonymous Credential Operations (time in ms), $n$ is attribute count

| $n$ | [ASM06] | [CDL16] | [PS16] | [TBA$^+$22] 2.2 | Our Improved 2.2 |
|---|---|---|---|---|---|
| | | | **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** |

# Chapter 7

# Conclusion and Future Work

This thesis delivers a suite of fast, secure, and expressive tools for digital identity systems. By achieving Show+Verify times as low as 3.08 ms, securing against malicious issuers, and enabling multi-credential proofs in 72 ms, I've shown that the privacy-security-usability trilemma can be resolved. T-SIRIS and efficient nullifiers further enhance scalability and trust, while the Rust benchmarking library provides a foundation for future research.

## 7.1 Summary

This thesis advances the security–privacy–usability trilemma in digital identity systems through five core contributions:

– **Chapter 2: Fast and Expressive Anonymous Credentials.** Introduced a new rerandomizable signature over commitments that reduces Show+Verify latency to 3.08 ms, secures against malicious issuers via mandatory key-verification protocols, and demonstrates sublinear expressive proofs using $\Sigma$–protocol optimizations.

– **Chapter 3: Identity Binding for Multi-Issuer, Multi-Credential Proofs.** Formalized the Identity Binding property, enabling secure linkage of up to 16 heterogeneous credentials in a single proof at 72 ms, with optional aggregation reducing verification to 31 ms.

– **Chapter 4: Efficient Nullifiers from q-DDHI.** Developed pairing-free VRFs and three novel zero-knowledge $\Sigma$-protocols to construct deterministic and rerandomizable nullifiers that achieve $5\times$ speedups over prior work.

– **Chapter 5: T-SIRIS — Threshold-Issued, Sybil-Resistant Identity System.** Designed and benchmarked a threshold issuance scheme that maintains near-constant Show+Verify times across increasing issuers, outperforming state-of-the-art threshold systems by up to 30x.

– **Chapter 6: Open-Source Benchmark Library.** Delivered a Rust library with standardized microbenchmarks for anonymous credential schemes, revealing practical gains from multi-scalar multiplication and optimized pairing routines.

## 7.2 Future Work

Key directions to extend this research include:

– *Post-Quantum Instantiations:* Explore lattice-based or hash-based analogues of rerandomizable signatures and nullifiers to anticipate quantum threats.

– *Rich Predicate Support:* Integrate circuit-level SNARKs or bulletproofs to enable arbitrary boolean and threshold predicates without sacrificing sub-10 ms verification.

– *Real-World Integration:* Prototype these primitives within production digital wallets (e.g., mobile SDKs, browser extensions) and evaluate end-user latency, compatibility, and developer ergonomics.

– *Dynamic Revocation and Auditing:* Design privacy-preserving revocation mechanisms that support both immediate revocation checks and retrospective audits, balancing unlinkability with accountability.

- *Adaptive Threshold Policies:* Investigate schemes where the issuance threshold adapts to context (e.g., higher thresholds for high-value credentials) without re-issuing keys.
- *Hardware Acceleration:* Leverage secure enclaves or GPU/FPGA offloading to further reduce proof generation and verification costs on edge devices.
- *Usability Studies:* Conduct human-centered evaluations to quantify the impact of sub-100 ms verifications on real-world user experience and adoption metrics.

## 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.

BBB$^+$17. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More, 2017. Publication info: Published elsewhere. Minor revision. 39th IEEE Symposium on Security and Privacy 2018.

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.

CKP$^+$23. E. Crites, M. Kohlweiss, B. Preneel, M. Sedaghat, and D. Slamanig. Threshold Structure-Preserving Signatures. In *Advances in Cryptology – ASIACRYPT 2023*, pages 348–382. Springer Nature Singapore, Singapore, 2023. 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.

CL04. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. *Advances in Cryptology – CRYPTO 2004*, 3152:56–72, 2004.

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.

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.

Dis24. Distributed Lab - Noir. noir-plume/BENCHMARK.md at main · distributed-lab/noir-plume, November 2024.

DKL$^+$23. J. Doerner, Y. Kondi, E. Lee, a. shelat, and L. Tyner. Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance, 2023. Great definition of anonymous credentials.

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.

EG14. A. Escala and J. Groth. Fine-Tuning Groth-Sahai Proofs. In *Public-Key Cryptography – PKC 2014*, pages 630–649, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

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.

GGM14. C. Garman, M. Green, and I. Miers. Decentralized Anonymous Credentials. In *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA, 2014. Internet Society.

Gro16. J. Groth. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. Series Title: Lecture Notes in Computer Science.

GS08. J. Groth and A. Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Advances in Cryptology – EUROCRYPT 2008*, pages 415–432, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

HNY$^+$25. L. Harchandani, O. Nosov, D. Yamamoto, gohan, and mark moir. Dock network crypto library, 2025.

HP22. C. Hébant and D. Pointcheval. Traceable Constant-Size Multi-authority Credentials. In *Security and Cryptography for Networks*, pages 411–434. Springer International Publishing, Cham, 2022. Series Title: Lecture Notes in Computer Science.

Int25. Internet Identity Workshop. Internet Identity Workshop, April 2025.

Jak09. Jakob Nielsen. Powers of 10: Time Scales in User Experience, October 2009.

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.

KPW15. E. Kiltz, J. Pan, and H. Wee. Structure-Preserving Signatures from Standard Assumptions, Revisited, 2015. Publication info: A major revision of an IACR publication in CRYPTO 2015.

Lin06. G. Linden. Geeking with Greg: Marissa Mayer at Web 2.0, November 2006.

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.

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].

SFA23. S. M. Sedaghat, Foteini Baldimtsi, and Arnab Roy. Groth-Sahai proofs: Zero to Hero!, June 2023.

Sig22. Signicat. The Battle to Onboard, 2022.

TBA$^+$22. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. UTT: Decentralized Ecash with Accountable Privacy. 2022.

TZ23. S. Tessaro and C. Zhu. Revisiting BBS Signatures, 2023. Publication info: A minor revision of an IACR publication in EUROCRYPT 2023.

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.

WGW$^+$23. K. Wang, J. Gao, Q. Wang, J. Zhang, Y. Li, Z. Guan, and Z. Chen. Hades: Practical Decentralized Identity with Full Accountability and Fine-grained Sybil-resistance. In *Annual Computer Security Applications Conference*, pages 216–228, Austin TX USA, December 2023. ACM.

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].