# Privacy-Preserving Identity Systems

No Author Given

No Institute Given

# Table of Contents

## 1 Introduction

Digital Identity systems form the foundation of online trust and authentication, processing billions of verifications daily [noa21, PCB+]. Traditional centralized systems, while effective for regulatory compliance [Elt24], suffer from significant privacy and security vulnerabilities. Ongoing data breaches [ZYD+22] affect billions of users, demonstrating the risks of storing identity data and centralizing systems. Decentralized Identity (DID) is a loosely defined framework (W3C specification) having evolved over the past 2 decades, offering users greater control over their credentials [SNA21], many concrete implementations struggle to balance privacy with accountability [MMZ+21] for the required feature set.

Anonymous Credential Systems (ACS) [Cha85, CL04, ASM06, PS16, FHS19] address privacy concerns but face challenges balancing privacy with accountability, orthogonally, unconditionally anonymous payment systems have demonstrated how unconditional anonymity can enable system abuse. Current systems focus on protecting against sybil attacks [CKS24, RARM], enabling revocation [CL02, CDR16, CDH16, BCD+17], rich attribute-based credential authentication [RWGM22, BS23] but very few implement all. The tension between privacy and accountability has become increasingly critical as governments worldwide, particularly the EU's Digital Identity Framework [noa24], move toward privacy-preserving digital identity wallets. These challenges motivate the need for a comprehensive identity system that achieves privacy, accountability, and practical deployment requirements simultaneously.

### 1.1 Organization

Short Summary: We build a new anonymous credential building block for multi-issuer, multi-credential systems. We use it to build a new private digital identity system with sybil resistance and revocation.

1. section 7 is the new anonymous credential system building block for multi-issuer, multi-credential
2. section **??** outlines all sigma zkp's used
3. section 14 builds a private identity system from the building block in section 7 and proofs in **??**
4. section 15 shows privacy and sybil resistance can be low-overhead, also shows our benchmarks for proving complex statements about credentials is more efficient than SOTA

### 1.2 Organization Update 8/2

We build a new anonymous credential building block for multi-issuer, multi-credential systems with an improved credential and sigma proof for sybil resistance.

Outline and Contributions

- 3 Outline the variant of Pedersen Commitments we use, we prove position binding in SDLP (first to do that)
- 4 Outline the variant of PS signatures we use, we prove security in AGM (best detailed proof) and improve verification efficiency and proof size compared to SOTA. We show why it's the best for complex use cases
- **??** Outline sigma protocols we use in the anonymous credential system, create the inverse protocol to prove hidden VRF
- 7 use the above 3 in a new multishow multiissuer anonymous credential system with sybil resistance
- 15 Analysis of the SOTA pairing based anonymous credentials

### 1.3 Related Work

**Decentralized Identity (DID)** enables entities to create and manage digital identities without relying on a central authority. W3C specifications for DID and Verifiable Credentials define standards for globally unique, publicly verifiable credentials, allowing a user to prove claims (information) about their identity attributes. DID typically uses Distributed Ledgers and public key cryptography to establish a "web of trust" and maintain revocation registries.

The DID model consists of

1. **Holders** Identities with Decentralized Identifiers (DID's) who manage their own keys, and credentials, and request access to resources
2. **Issuers** create and sign credentials about identity holders
3. **Verifiers** validate credentials by checking the presented cryptographic information against the registry
4. **Verifiable Data Registry**, often a DLT, is the root of trust, maintaining DID records, keys, and credential schemas, but doesn't store credential data
5. **Identity Wallet**: the user interface for storing, managing, and presenting verifiable credentials

Citations - U-Prove, U-Port, Connect.me, Sovrin, PingID, w3c

While the W3C DID specifications outline core functionalities such as cryptographic verification, privacy preservation, selective disclosure, and revocation, it requires a formal security definition and proofs to achieve these properties. Anonymous Credential Systems provide well-established formal security definitions for many properties that DID aims to achieve. Specifically, Correctness, Unforgeability, Anonymity, Sybil Resistance, and Revocation. By building DID systems on top of Anonymous Credential primitives, DID systems can inherit these formal security guarantees.

**Multi Attribute Anonymous Credentials** Anonymous Credentials are a long line of orthogonal work with the goal of providing privacy/anonymity to online interactions, These are now deployed in a number of real-world systems (U-Prove, Idemix, PrivacyPass). The line of work has improved with features, efficiency, and expressiveness throughout time. Abstractly, they provide private protocols to prove information on committed attributes. Pairing-based blind signatures enabled efficient multi-show credentials (CL, PS, BBS+). Other constructions are based on (chase's MAC's, etc) Further work was done to the Anonymous Credential primitive to support delegation, update, and pseudonyms and may also require incorporating other cryptographic primitives to enable revocation, auditing, tracing, and sybil resistance.

**Anonymous Credential Systems (ACS)** Anonymous Credential Systems implement primitives together in ways that preserve privacy and offer additional functionality required by systems. The combination of multiple primtiives to be used together in a privacy preserving way is complex it itself.

**Accountable Privacy** In a parallel industry, fully private financial systems like zcash, monero, and tornado cash where blockchain users can exploit and misuse the system is a big problem and will prevent governments and organisations to deploy private preserving techniques. Innovations in Identity often follow those in financial systems, (Chaum's e-cash and blind signatures, DAC and ZCash for example) as they share similar motivations and infrastructure, spending a coin anonymously is analogous to using an identity anonymously. Therefore, advancements in the field of Accountable Privacy (UTT, other examples), a series of techniques and protocols that balance the tradeoffs between privacy and accountability in financial systems, can inspire and motivate private identity systems.

**Privacy Preserving Decentralized Identity Systems** CanDID proposed a privacy-preserving decentralized identity system achieving Sybil Resistance, Accountability. They defined a system architecture that has been extended in many different directions, such as [WGW+23,RPX+22] optimising for blockchain performance and additional private-accountability features. [CKS24, RARM] optimize for non-interactivity and minimal stored state with the CanDID threshold committee

CanDID leaves integrating Decentralized Identity with Anonymous Credentials as open work.

**Decentralized Identity with Anonymous Credentials** The strawman approach to combining Decentralized Identity with Anonymous Credentials replaces the existing credential, usually in the form of a signature over attributes, signed by an identity provider and verifiable with their public key, with a blind signature over attributes, which introduces the accountable privacy problems such as how to issue and verify a credential to an anonymous user, and how to retain accountability of an anonymous user, such are the real-world identity system requirements. Compatibility for a decentralized architecture requires decentralized cryptography which has propelled the use of threshold cryptography in anonymous credential schemes and other settings e.g. (Coconut, Threshold BBS+, ...).

### 1.4 Contributions

We present a privacy-preserving decentralized identity system for multi-issuer environments. Our system combines anonymous credential primitives with decentralized identity architecture, achieving security and privacy properties that are challenging to realize when naively combining these building blocks. Our main contributions are:

1. A privacy-preserving identity system that enables secure credential chaining and complex anonymous identity verification across multiple issuers.

2. A novel zero-knowledge building block that enables private proofs of VRF derivations from committed messages.

3. A complete system implementation using PS Signatures over commitments that achieves sybil resistance and revocation while enabling multi-issuer credential chaining - validated through concrete implementations, benchmarks, and formal security analysis.

### 1.5 Gap Analysis

Digital identity is undergoing a fundamental transformation, evolving across three frontiers: decentralization, mandatory institutional adoption, and the emergence of attestation services. Identity systems are evolving from trusted, single-issuer models where a user authenticates with a single authority toward a decentralized paradigm where users publicly verify any multitude of credentials, manage multiple credentials for diverse issuers with their digital wallets.

As traditional organizations increasingly adopt decentralized identity capabilities and while it's also being mandated in the EU, they seek solutions that minimize changes to their existing infrastructure while enabling new DID capabilities and maintaining regulatory compliance. Beyond traditional organisations, a new frontier of credential issuance is emerging through automated attestation services like TLS Notary, Chainlink's DECO, Brave Browser's distefano, and Sui Labs zkLogin which enable verifiable data to become a credential. This transformation, while powerful, introduces challenges to identity systems run by governments and trusted organisations who require sybil resistance protecation and revocation while maintaining privacy in a system where traditional infrastructure assumptions such as ease of revocation no longer hold.

**Core Challenges** Evolving from single-issuer to multi-issuer, multi-credential environments introduces several challenges. While existing solutions support private identity systems with anonymity, sybil resistance and revocation for single issuers. The introduction of multiple credentials and their sources transforms solved problems into new challenges. A decentralized system for the frontier of credentials must maintain anonymity across credential presentations, implement cross credential sybil resistance and efficient revocation checks without centralized trust.

Additionally, composing privacy-preserving primitives together to achieve the properties we require introduces complexity. While individual primitives for anonymous credentials, Sybil resistance, and revocation are well understood in isolation, the integration highlights the trilemma of accountable privacy systems - the tension between privacy, accountability, and functionality. The core challenge lies in designing efficient zero knowledge proof systems that combine these primitives in protocols that maintain the security and privacy properties of our system with practical efficiency.

Thirdly arises when users verify attributes from multiple credentials. Secure credential composition is required, while allowing flexible zero knowledge proof attribute attestations and selective disclosure. Lastly, users with multiple credentials require to privately prove their credentials are not revoked, introducing a scaling challenge - enabling efficient zero-knowledge batch proofs of non-membership while maintaining privacy and practical verification times.

### 1.6 Technical Challenges

Building a privacy-preserving decentralized identity system requires balancing competing requirements: adhereing to strong security and privacy properties while retaining accountability measures and providing efficient verification of complex identity statements. While individual cryptographic primitives exist for many of these properties in isolation, combining them while maintaining security and efficiency introduces technical challenges, we identity three fundamental challenges below:

Sam: Rewrite this - start with rerand sigs over commitments, then extending that for multi-issuer, multi-cred, then using that for identity system

1. **Efficient Rerandomizable Signatures over Commitments** A key technical challenge was designing a signature scheme that efficiently supports both rerandomization and zero-knowledge proofs over-committed attributes. While existing schemes like BBS+, CL, and standard PS provide these properties, we use a customized PS signature with the lowest overhead in the randomization step. Unlike BBS+ and CL04, we maintain compatibility with standard Pedersen Commitments, enabling efficient proofs from standard techniques in the literature.

2. **Sybil-Resistant Context Credential Construction** Designing an efficient mechanism to link context credentials to a master credential while preserving privacy, our solution uses a novel building block that combines a VRF with committed attributes - the user's Master Credential contains a commitment with their VRF key and generates a context credential nullifier with a VRF parameterized by the key and input the context string. The complexity lies in efficiently proving in zero knowledge this nullifier was correctly derived from the committed key present in a valid, unrevoked master credential. This construction enables strong sybil resistance while maintaining unlinkability between presentations.

3. **Efficient Multi-Credential Proofs and Revocation** enabling efficient proofs over multiple credentials while ensuring practical revocation. Our construction leverages Sigma protocols and Pedersen commitments, which, although they scale linearly with the credential attributes, they are extremely efficient in practice and support the most expressive statements. We integrate existing efficient revocation mechanisms that support batch non-membership proofs, allowing multiple credentials to be efficiently verified simultaneously while maintaining anonymity through zero-knowledge proof protocols.

Table 1: Comparison of our construction over previous work.

| Features | Multi Issuer | Sybil Resistance | Revocation | Efficient Cred. Chaining[1] | M-ABC[2] | Anonymity[3] |
|---|---|---|---|---|---|---|
| CanDID [MMZ+21] | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| SyRA [CKS24] | ✗ | ✓ | ✗ | ✗ | ✗ | ✓ |
| S3ID [RARM] | ✓ | ✓ | ✗ | ✓ | ✗[4] | ✓ |
| Our Work | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

[1] Credential Chaining is a user presenting multiple credentials to be verified together for a complex identity statement.
[2] M-ABC is a Multi-Show Attribute Based Credential, allowing a user to satisfy rich, attribute-based identity statements
[3] Anonymity is defined in the Anonymous Credential model, no verifier and issuer (collaborating together) may learn more about the user or their credentials other than what the user discloses and what their credentials verify. Multiple credential verifications are unlinkable.
[4] While possible in S3ID, they mention
[5] Multi-issuer means supporting credentials from different authorities that can be cryptographically linked while preserving privacy

## Comparison

Sam: S3ID is inefficient for attribute-based verification, this table doesn't show that

## 2 Preliminaries

### 2.1 Notation

Let $\mathbb{N}$ is the set of positive integers $\{1, 2, 3, \ldots\}$. If $x$ is a string, then $|x|$ is its length, if $S$ is a set, $|S|$ denotes its size. If $\lambda \in \mathbb{N}$, then $1^\lambda$ denotes the string of $\lambda$ ones. If $n$ is an integer, then $[n] = \{1, \ldots, n\}$. If $S$ is a set, then $s \leftarrow\!\!\$\ S$ denotes the operation of picking an element $s$ of $S$ uniformly at random. \$ denotes randomized algorithm output. We use $z \leftarrow A()$ and $z = A()$ interchangably for deterministic assignment where $z$ is the output of a deterministic algorithm $A$ and $z \leftarrow A(x, y, \ldots)$ is the output of a deterministic algorithm with inputs $x, y, \ldots$ when the set, sum, or product has a single index. The notation $\vec{x}$ is used to denote the vector $(x_0, \ldots, x_n)$ or $(x_1, \ldots, x_n)$ where $n$ and choice of $0, 1$ will be clear from the context.

The notation $\{x_i\}_1^n, \sum_1^n x_i, \prod_1^n x_i$ are shorthand for $\{x_i\}_{i=1}^n, \sum_{i=1}^n x_i$ and $\prod_{i=1}^n x_i$ respectively.

We write $A(x, y, \ldots : \mathsf{O}_1, \mathsf{O}_2, \ldots)$ as an algorithm taking $(x, y, \ldots)$ as input with access to oracles $(\mathsf{O}_1, \mathsf{O}_2, \ldots)$ and $z \leftarrow\!\!\$\ A(x, y, \ldots : \mathsf{O}_1, \mathsf{O}_2, \ldots)$ as the assignment of the output of $A$ to $z$. We denote the adversary algorithm as $\mathcal{A}$

We write commitments using 1-base indexing. $\mathsf{cm}_m[1]$ represents the message at the first position of the $\mathsf{cm}_m$ commitment, that is, in a commitment $\mathsf{cm}_m = CM.Com([s, k, \ldots]; r)$, algebraically, the commitment is $g_1^s g_2^k \ldots h^r$ where $s$ is the first position.

Use this to update mine [BCN+10] has good preliminaries

$\mathbb{Z}_p^*, \mathbb{G}_1^*$ - the star (*) represents the exclusion of trivial elements. That is, $\mathbb{Z}_p^*$ excludes 0, focusing on invertible (coprime with $p$) elements under multiplication mod $p$ where $|\mathbb{Z}_p^*| = p - 1$ and sampling $y \leftarrow\!\!\$\ \mathbb{Z}_p^*$ selects an integer from $[1, p-1]$ uniformly. $\mathbb{Z}_p$ is the additive group mod $p$, size $=$ $|\mathbb{Z}_p| = p$, sampling $\leftarrow\!\!\$\ \mathbb{Z}_p$ samples uniformly from $[0, p-1]$

$\mathbb{G}_1^*$ excludes the elliptic curve identity element $\{\mathcal{O}\}$ required for discrete log operations as any elliptic curve operation with the identity element results in the identity element.

For a probabilistic algorithm $A$, we write $x \leftarrow A(y_1, \ldots, y_n)$ to denote sampling $x$ according to the distribution induced by $A$'s random coins on inputs $y_1, \ldots, y_n$.

For an interactive protocol $P = (P_0, P_1)$ with inputs $in_0, in_1$ and outputs $out_0, out_1$, we write $\langle out_0; out_1 \rangle \leftarrow \langle P_0(in_0); P_1(in_1) \rangle$ to denote protocol execution.

For an array $\mathsf{arr}$, we denote its $i$th element by $\mathsf{arr}[i]$.

**Definition 1 (Bilinear map).** *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order $p$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive and $\mathbb{G}_T$ is multiplicative. Let $g$ and $\tilde{g}$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map or pairing if it is efficiently computable and the following holds:*

- ***Bilinearity:*** $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab} = e(g^b, \tilde{g}^a)\ \forall a, b \in \mathbb{Z}_p$.
- ***Non-degeneracy:*** $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$*, i.e.,* $e(g, \tilde{g})$ *generates* $\mathbb{G}_T$.

*If $\mathbb{G}_1 = \mathbb{G}_2$, then $e$ is symmetric (Type-1) and asymmetric (Type-2 or 3) otherwise. For Type-2 pairings, there is an efficiently computable isomorphism $\Psi : \mathbb{G}_2 \to \mathbb{G}_1$ but none from $\mathbb{G}_1 \to \mathbb{G}_2$; for Type-3 pairings, no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known. In practice, we use Type-3 pairings as they provide the best balance between security and efficiency at standard security levels.*

**Definition 2 (Bilinear Group Generator).** *A bilinear group generator* $\mathsf{BGGen}$ *is a probabilistic polynomial-time algorithm that on input security parameter $1^\lambda$, outputs a tuple* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g})$ *where:*

- *$p$ is a prime of bit-length $\lambda$*
- *$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are cyclic groups of order $p$*
- *$g$ and $\tilde{g}$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively*
- *$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate bilinear map*

**Definition 3 (Discrete Logarithm in Bilinear Groups).** *For a bilinear group generator* $\mathsf{BGGen}$, *we say the discrete logarithm assumption holds if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

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

*The assumption is analogously defined for $\mathbb{G}_2$ using $\tilde{g}^a$.*

**Definition 4 (LRSW Assumption in Bilinear Groups).** *For any PPT adversary $\mathcal{A}$, we say the LRSW assumption holds if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{l} \mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\$ \ \mathsf{BGGen}(1^\lambda) \\ x, y \leftarrow\!\$ \ \mathbb{Z}_p, X \leftarrow g^x, \tilde{Y} \leftarrow \tilde{g}^y \\ (m^*, A, B, C) \leftarrow\!\$ \ \mathcal{A}^{\mathcal{O}_{x,y}}(\mathsf{BG}, X, \tilde{Y}) \end{array} : \begin{array}{l} m^* \notin Q \quad \wedge \quad \exists r \in \mathbb{Z}_p : \\ A = g^r \wedge B = A^y \wedge C = A^{x+m^*xy} \end{array} \right] \leq \mathsf{negl}(\lambda)(\lambda)$$

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

$$\textit{Oracle } \mathcal{O}_{x,y}(m) : \ \textit{Samples } r \leftarrow\!\$ \ \mathbb{Z}_p, \ \textit{Returns } (g^r, (g^r)^y, (g^r)^{x+mxy})$$

**Definition 5 (PS-LRSW Assumption in Bilinear Groups).** *For any PPT adversary $\mathcal{A}$, we say the PS-LRSW assumption holds if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

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

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

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

**Definition 6 (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* $\mathsf{negl}(\lambda)$ *such that:*

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

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

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

*Furthermore, in the generic group model after $q$ oracle queries and $q_G$ group-oracle queries, the probability of success is bounded by $\mathcal{O}((q + q_G)^2/p)$.*

**Definition 7 (XDDH/SXDH Assumption in Bilinear Groups).** *For any PPT adversary $\mathcal{A}$, we say the XDDH assumption holds in $\mathbb{G}_i$ ($i \in \{1,2\}$) if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\left| \Pr\begin{bmatrix} \mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\!\$\ \mathsf{BGGen}(1^\lambda) \\ a, b \leftarrow\!\!\$\ \mathbb{Z}_p, c \leftarrow ab \bmod p \\ d \leftarrow\!\!\$\ \mathcal{A}(\mathsf{BG}, g_i^a, g_i^b, g_i^c) \end{bmatrix} - \Pr\begin{bmatrix} \mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\!\$\ \mathsf{BGGen}(1^\lambda) \\ a, b, r \leftarrow\!\!\$\ \mathbb{Z}_p \\ d \leftarrow\!\!\$\ \mathcal{A}(\mathsf{BG}, g_i^a, g_i^b, g_i^r) \end{bmatrix} \right| \leq \mathsf{negl}(\lambda)$$

*where $g_i$ is the generator of $\mathbb{G}_i$. The SXDH assumption holds if XDDH holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

**Definition 8 (Symmetric Discrete Logarithm Assumption (SDLP)).** *For any PPT adversary $\mathcal{A}$, we say the SDLP assumption holds if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

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

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

Assumptions from here - as discussed by UTT paper commitment instantiation The binding property holds under the SDL assumption of [BCN+10] which follows from 1-SDH of [BB08]. (See Section I for details.)

## 2.2 Digital Signatures

**Definition 9 (Signature Scheme).** *A signature scheme* $\mathsf{Sig}$ *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *where:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$*: is a probabilistic algorithm that takes as input a security parameter $\lambda$ and outputs a secret signing key* $\mathsf{sk}$ *and a public verification key* $\mathsf{pk}$*. The message space $\mathcal{M}$ is implicitly defined by* $\mathsf{pk}$.
- $\mathsf{Sign}(\mathsf{sk}, m; r) \to \sigma$*: is a probabilistic algorithm that takes as input the secret key* $\mathsf{sk}$*, 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* $\mathsf{pk}$*, 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.*

*The scheme must satisfy the following correctness property: For all $\lambda \in \mathbb{N}$, all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, all messages $m \in \mathcal{M}$, and all random coins $r \in \mathcal{R}$:*

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

**Definition 10 (Correctness).** *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is correct if for all $k \in \mathbb{N}$, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{KeyGen}(1^k)]$ *and all $m \in \mathcal{M}$ we have:*

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

**Definition 11 (EUF-CMA Security).** *A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is existentially unforgeable under adaptive chosen-message attacks if for all PPT algorithms $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

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

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

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

## 3 Symmetric Rerandomizable Vector Commitments

We now introduce a specialized Pedersen commitment scheme from [TBA+22] and prove tight security as the building block for our anonymous credential system. Our construction extends the classical Pedersen Commitment scheme with three properties that are required for our multi-issuer, multi-credential setting construction:

1. **Position Binding:** we need to commit to a vector of attributes $\{m_1, \ldots, m_n\}$ while ensuring the individual positions within the commitment remain binding. Position binding is useful when proving relations about specific committed attributes across multiple credentials.

2. **Symmetric across bilinear groups:** in order to support our pairing-based signatures and efficient zero-knowledge proofs, we require commitments to be represented and verified in both $\mathbb{G}_1, \mathbb{G}_2$ curve groups.

3. **Rerandomization:** we need the ability to rerandomize commitments while preserving their structure which supports unlinkability in credential presentations

Furthermore, our scheme often leverages Pedersen Commitments homomorphic properties, allowing us to combine multiple commitments with group operations. This enables us to bind credential attributes and hide secrets while maintaining privacy.

**Definition**

**Definition 12 (Rerandomizable Vector Commitment scheme).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is a tuple* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Rerand})$ *of PPT algorithms where:*

– $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{ck}$ *probabilistic algorithm that takes as input the security parameter $\lambda$ and vector length $n$ both in unary. It outputs the commitment key* $\mathsf{ck}$ *of length $n$*

– $\mathsf{Commit}(\mathsf{ck}, m) \to (\mathsf{cm}, r)$*: probabilistic algorithm takes as input the commitment key* $\mathsf{ck}$ *and message vector $m \in \mathcal{M}^n$ . Outputs the commitment* $\mathsf{cm}$ *and opening key $r$*

– $\mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \vec{m}, r) \to \{0,1\}$*: deterministic algorithm, takes as input the commitment key* $\mathsf{ck}$*, commitment* $\mathsf{cm}$*, messages $\vec{m}$ and randomness $r$, outputs 1 if success, 0 for failure*

– $\mathsf{Rerand}(\mathsf{ck}, \mathsf{cm}) \to (cm', r')$*: probabilistic algorithm, takes as input the commitment key* $\mathsf{ck}$*, commitment* $\mathsf{cm}$ *outputs randomized commitment* $\mathsf{cm}'$ *and new randomness $r'$.*

A rerandomizable vector commitment scheme is secure if it's correct, hiding, binding, position binding, rerandomizable, and symmetric group correct. We use standard definitions for correctness, hiding, and binding.

**Definition 13 (Correctness).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is correct if for all $n \in \mathbb{N}$ and all $\vec{m} \in \mathcal{M}^n$:*

$$\Pr\left[\begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{cm}, r) \leftarrow \mathsf{Commit}(\mathsf{ck}, \vec{m})\end{matrix} : 1 \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \vec{m}, r)\right] = 1$$

**Definition 14 (Perfect Hiding).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is perfectly hiding if for all (unbounded) adversaries $\mathcal{A}$ and all message vectors $\vec{m}_0, \vec{m}_1$ of equal length:*

$$\Pr\left[\begin{matrix}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\vec{m}_0, \vec{m}_1) \leftarrow \mathcal{A}(\mathsf{ck}) \\ b \leftarrow\$ \{0,1\} \\ (\mathsf{cm}_b, r_b) \leftarrow \mathsf{Commit}(\mathsf{ck}, \vec{m}_b) \\ b' \leftarrow \mathcal{A}(\mathsf{ck}, \mathsf{cm}_b)\end{matrix} : b' = b\right] = \frac{1}{2}$$

**Definition 15 (Binding).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is binding if for all* $\mathsf{PPT}$ *adversaries* $\mathcal{A}$ *and all message vectors* $\vec{m}_0, \vec{m}_1$ *of equal length, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)\\ (\vec{m}_0, \vec{m}_1, r_0, r_1) \leftarrow \mathcal{A}(\mathsf{ck})\\ \mathsf{cm}_0 \leftarrow \mathcal{A}(\mathsf{CM.Com}(\mathsf{ck}, \vec{m}_0, r_0))\\ b_0 \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \vec{m}_0, r_0)\\ b_1 \leftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{cm}, \vec{m}_1, r_1)\end{array} \quad : \quad b_0 = b_1 = 1 \quad \wedge \quad \vec{m}_0 \neq \vec{m}_1\right] \leq \mathsf{negl}(\lambda)$$

*Informally, No adversary* $\mathcal{A}$ *should be able to open a commitment to two different messages, even when allowed to use different randomness values. This captures the "unforgeability" aspect of commitments.*

**Definition 16 (Position Binding).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is position binding if for all* $\mathsf{PPT}$ *adversaries* $\mathcal{A}$ *and where* $\vec{m}_0, \vec{m}_1$ *are vectors that differ only at position i, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

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

*Informally, this states that an adversary* $\mathcal{A}$ *should not be able to open a commitment* $\mathsf{cm}$ *to index i unless both commitments use the same message, even when the commitments only differ by position i*

**Definition 17 (Symmetric Group Correctness).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *satisfies symmetric group correctness if for all* $n \in \mathbb{N}$ *and all* $\vec{m} \in \mathcal{M}^n$*:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow\$ \mathsf{Setup}(1^\lambda, 1^n)\\ (\mathsf{cm}, \widehat{\mathsf{cm}}, r) \leftarrow\$ \mathsf{Commit}(\mathsf{ck}, \vec{m})\end{array} : e(\mathsf{cm}, \widehat{g}) = e(g, \widehat{cm})\right] = 1$$

$$\Pr\left[(\mathsf{cm}', \widehat{\mathsf{cm}'}, r') \leftarrow\$ \mathsf{Rerand}(\mathsf{ck}, \mathsf{cm}, \widehat{\mathsf{cm}}, r) \ : e(\mathsf{cm}', \widehat{g}) = e(g, \widehat{\mathsf{cm}'})\right] = 1$$

*where* $g, \widehat{g}$ *are the generators specified in* $\mathsf{ck}$ *for* $\mathbb{G}_1, \mathbb{G}_2$ *respectively.*

**Definition 18 (Additive Homomorphism).** *A rerandomizable vector commitment scheme* $\mathsf{RVC}$ *is additively homomorphic if for all* $\vec{m}_1, \vec{m}_2 \in \mathcal{M}^n$ *and randomness* $r_1, r_2 \in \mathbb{Z}_p$*:*

$$\Pr\left[\begin{array}{l}\mathsf{ck} \leftarrow\$ \mathsf{Setup}(1^\lambda, 1^n)\\ (\mathsf{cm}_1, r_1) \leftarrow\$ \mathsf{Commit}(\mathsf{ck}, \vec{m}_1)\\ (\mathsf{cm}_2, r_2) \leftarrow\$ \mathsf{Commit}(\mathsf{ck}, \vec{m}_2)\end{array} : \mathsf{cm}_1 \cdot \mathsf{cm}_2 = \mathsf{Commit}(\mathsf{ck}, \vec{m}_1 + \vec{m}_2, r_1 + r_2)\right] = 1$$

*Furthermore, for symmetric group correctness, the homomorphism must hold across groups:*

$$e(\mathsf{cm}_1 \cdot \mathsf{cm}_2, \widehat{g}) = e(g, \widehat{\mathsf{cm}_1} \cdot \widehat{\mathsf{cm}_2})$$

## 3.1 Construction

We instantiate a dual-group Pedersen Vector Commitment scheme with groups $\mathbb{G}_1, \mathbb{G}_2$ to enable efficient verification within our signature construction. For commitments $\mathsf{cm} \in \mathbb{G}_1, \widetilde{\mathsf{cm}} \in \mathbb{G}_2$, we verify consistency via the pairing relation $e(\mathsf{cm}, \tilde{g}) = e(g, \widetilde{\mathsf{cm}})$.

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order $p$ with an efficient Type-3 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. For message vector $\vec{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}_p^\ell$, our rerandomizable commitment scheme consists of the following algorithms:

– $\mathsf{CM.Setup}(1^\lambda, \ell) \to \mathsf{ck}$: Sample generators $(g, \tilde{g}) \leftarrow\$ \mathbb{G}_1 \times \mathbb{G}_2$ For $i \in [1, \ell]$: Sample $y_i \leftarrow\$ \mathbb{Z}_p$, compute $(g_i, \tilde{g}_i) \leftarrow (g^{y_i}, \tilde{g}^{y_i})$ Return $\mathsf{ck} \leftarrow (g, (g_1, \ldots, g_\ell), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_\ell))$

- $\mathsf{CM.Com}(\mathsf{ck}, \vec{m}) \to (\mathsf{cm}, \widetilde{\mathsf{cm}}, r)$: Parse $\mathsf{ck}$ as $(g, (g_1, \dots, g_\ell), \tilde{g}, (\tilde{g}_1, \dots, \tilde{g}_\ell))$ Sample $r \leftarrow\!\$\ \mathbb{Z}_p$ Compute $\mathsf{cm} \leftarrow g^r \prod_{i=1}^{\ell} g_i^{m_i}$ and $\widetilde{\mathsf{cm}} \leftarrow \tilde{g}^r \prod_{i=1}^{\ell} \tilde{g}_i^{m_i}$ Return $(\mathsf{cm}, \widetilde{\mathsf{cm}}, r)$
- $\mathsf{CM.Rerand}(\mathsf{ck}, \mathsf{cm}, \widetilde{\mathsf{cm}}, r_\Delta) \to (\mathsf{cm}', \widetilde{\mathsf{cm}'})$: Parse $\mathsf{ck}$ as $(g, \cdot, \tilde{g}, \cdot)$ Compute $\mathsf{cm}' \leftarrow \mathsf{cm} \cdot g^{r_\Delta}$ and $\widetilde{\mathsf{cm}'} \leftarrow \widetilde{\mathsf{cm}} \cdot \tilde{g}^{r_\Delta}$ Return $(\mathsf{cm}', \widetilde{\mathsf{cm}'})$
- $\mathsf{CM.Open}$

*Homomorphic Properties* The scheme inherits the additive homomorphic properties of Pedersen commitments:

- **Rerandomization:** For commitment $\mathsf{cm} = g^r \prod_{i=1}^{\ell} g_i^{m_i}$ and fresh randomness $r_\Delta \leftarrow\!\$\ \mathbb{Z}_p$, the rerandomized commitment $\mathsf{cm}' = \mathsf{cm} \cdot g^{r_\Delta} = g^{r+r_\Delta} \prod_{i=1}^{\ell} g_i^{m_i}$ preserves the message vector while updating the randomness from $r$ to $r+r_\Delta$. By the DDH assumption in $\mathbb{G}_1$, $\mathsf{cm}'$ is computationally indistinguishable from $\mathsf{cm}$.
- **Two-Party Protocol:** For joint randomness generation:

  1. User samples $s_1, r \leftarrow\!\$\ \mathbb{Z}_p$ and commits: $\mathsf{cm}_1 \leftarrow g^r g_1^{s_1}$
  2. Authority samples $s_2 \leftarrow\!\$\ \mathbb{Z}_p$ and computes: $\mathsf{cm}_2 \leftarrow \mathsf{cm}_1 \cdot g_1^{s_2} = g^r g_1^{s_1+s_2}$
  3. Authority returns $(\mathsf{cm}_2, s_2)$, establishing shared secret $s_1 + s_2$

  Security holds bidirectionally:

  - **Unforgeability:** Computational binding under DL prevents malicious users from changing $s_1$ post-commitment
  - **Anonymity:** Perfect hiding of $\mathsf{cm}_1$ ensures zero knowledge of $s_1$ against malicious issuers

## 3.2 Security of our Construction

**Theorem 1.** *In the Algebraic Group Model, if the Symmetric Discrete Logarithm Problem (SDLP) is hard in the bilinear group* $\mathsf{BG}$*, then our RVC scheme satisfies position binding. Specifically, for any algebraic PPT adversary* $\mathcal{A}$ *against position binding, there exists a PPT reduction* $\mathcal{B}$ *against SDLP such that:*

$$\mathsf{Adv}^{\mathsf{pos\text{-}bind}}_{\mathcal{A},\mathsf{RVC}}(\lambda) \leq \ell \cdot \mathsf{Adv}^{\mathsf{SDLP}}_{\mathcal{B},\mathbb{G}}(\lambda)$$

*where* $\ell$ *is the vector length.*

*Proof.* We prove via reduction in the AGM. Given an algebraic PPT adversary $\mathcal{A}$ that breaks position binding with non-negligible probability $\epsilon$, we construct a PPT algorithm $\mathcal{B}$ that solves SDLP with probability $\epsilon/\ell$. For clarity, we illustrate with $\ell = 3$; the proof generalizes naturally.

Algorithm $\mathcal{B}$ works as follows:

1. **Setup**: On input SDLP instance $(g^x, \tilde{g}^x) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\mathcal{B}$ proceeds to:
   (a) Sample $i^* \leftarrow\!\$\ [1, \ell]$ uniformly at random
   (b) For position $i^*$: set $(g_{i^*}, \tilde{g}_{i^*}) \leftarrow (g^x, \tilde{g}^x)$
   (c) For positions $j \neq i^*$: sample $y_j \leftarrow\!\$\ \mathbb{Z}_p$, set $(g_j, \tilde{g}_j) \leftarrow (g^{y_j}, \tilde{g}^{y_j})$
   (d) Give $\mathsf{ck} = ((g_1, g_2, g_3), (\tilde{g}_1, \tilde{g}_2, \tilde{g}_3))$ to $\mathcal{A}$
2. **Position Binding Break**: Since $\mathcal{A}$ is algebraic, when it outputs $(\mathsf{cm}, i, \vec{m}_0, \vec{m}_1, r_0, r_1)$, it also provides the representation of $\mathsf{cm}$ in terms of the generators:

- $\mathsf{cm} \in \mathbb{G}_1$ with its algebraic representation
- $i \in [1, \ell]$ is the position where binding breaks
- $\vec{m}_0, \vec{m}_1 \in \mathbb{Z}_p^\ell$ differ only at position $i$
- $r_0, r_1 \in \mathbb{Z}_p$ are opening randomness values

3. **Extracting SDLP**: If $i \neq i^*$, abort. Otherwise:
   (a) By the algebraic property of $\mathcal{A}$, we have explicit representations of the commitment openings:
   $$g^{r_0} g_1^{m_{0,1}} g_2^{x \cdot m_{0,2}} g_3^{m_{0,3}} = g^{r_1} g_1^{m_{1,1}} g_2^{x \cdot m_{1,2}} g_3^{m_{1,3}}$$
   (b) Since these representations are explicit in the AGM, we can directly compare exponents:
   $$r_0 + y_1 m_{0,1} + x m_{0,2} + y_3 m_{0,3} = r_1 + y_1 m_{1,1} + x m_{1,2} + y_3 m_{1,3}$$
   (c) Since $\vec{m}_0$ and $\vec{m}_1$ differ only at position $i^* = 2$, we have $m_{0,1} = m_{1,1}$ and $m_{0,3} = m_{1,3}$. Terms cancel:
   $$r_0 + x m_{0,2} = r_1 + x m_{1,2}$$
   (d) Solve for $x$:
   $$x \equiv \frac{r_1 - r_0}{m_{0,2} - m_{1,2}} \pmod p$$
   Note: Division is well-defined as $m_{0,2} \neq m_{1,2}$ by assumption.

The reduction succeeds whenever $i = i^*$ and $\mathcal{A}$ succeeds, which occurs with probability $\epsilon/\ell$. This is non-negligible when $\epsilon$ is non-negligible, contradicting the SDLP assumption.

*Analysis:* We analyze the reduction's properties in detail:

- **Perfect Simulation:** The commitment key distribution is identical to the real scheme:
  - At position $i^*$: $(g_{i^*}, \tilde{g}_{i^*}) = (g^x, \tilde{g}^x)$ is uniformly distributed in $\mathbb{G}_1 \times \mathbb{G}_2$ by the SDLP instance properties
  - At positions $j \neq i^*$: $(g_j, \tilde{g}_j) = (g^{y_j}, \tilde{g}^{y_j})$ is uniform due to $y_j \leftarrow\!\!\$\ \mathbb{Z}_p$
  - Therefore, from $\mathcal{A}$'s view, $\mathsf{ck}$ is distributed identically to the real scheme
- **Extraction Success:** $\mathcal{B}$ successfully extracts the SDLP solution when:
  - $\mathcal{A}$ outputs a valid position binding break (occurs with probability $\epsilon$)
  - The guessed position matches: $i = i^*$ (occurs with probability $1/\ell$)
  - The extraction equation is solvable: $m_{0,i^*} \neq m_{1,i^*}$ (guaranteed by definition of position binding break)
- **Advantage Analysis:** Combining these probabilities:
  - Events are independent as $i^*$ is chosen before $\mathcal{A}$'s execution
  - $\mathsf{Pr}[\mathcal{B} \text{ succeeds}] = \epsilon \cdot \frac{1}{\ell}$
  - Therefore: $\mathsf{Adv}^{\mathsf{SDLP}}_{\mathcal{B},\mathbb{G}}(\lambda) \geq \frac{1}{\ell} \cdot \mathsf{Adv}^{\mathsf{pos\text{-}bind}}_{\mathcal{A},\mathsf{RVC}}(\lambda)$

Thus, if $\mathcal{A}$ breaks position binding with non-negligible probability $\epsilon$, then $\mathcal{B}$ solves SDLP with non-negligible probability $\epsilon/\ell$, contradicting the SDLP hardness assumption in $\mathbb{G}$.

## 4 Rerandomizable Signature over Commitments

### Definition

**Definition 19 (Rerandomizable Signature over Commitments).** *A rerandomizable signature scheme over commitments* $\mathsf{RS}$ *is a tuple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Rerand}, \mathsf{Ver}, \mathsf{VerKey})$ *of PPT algorithms where:*

- $\mathsf{RS.KeyGen}(\mathsf{pp}, \mathsf{ck}) \to (\mathsf{sk}, \mathsf{pk} = (\mathsf{pp}, \mathsf{vk}, \mathsf{ck}))$ *is a probabilistic algorithm that takes in the public parameters* $\mathsf{pp}$ *and commitment key* $\mathsf{ck}$*, outputs a signing key* $\mathsf{sk}$*, a public verification key* $\mathsf{vk}$ *and outputs* $(\mathsf{sk}, \mathsf{pk} = (\mathsf{pp}, \mathsf{vk}, \mathsf{ck}))$

- $\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{cm}; u) \to \sigma$*: probabilistic algorithm takes the signing key* $\mathsf{sk}$*, commitment* $\mathsf{cm}$ *from the commitment space* $\mathcal{C}$ *and random coins* $u$ *sampled from random space of the signature scheme. Output* $\sigma$

- $\mathsf{RS.Rerand}(\mathsf{pk}, \sigma, r_\Delta, u_\Delta) \to \sigma'$ *is a deterministic algorithm that enables signature rerandomization. Takes a public key* $\mathsf{pk} = (\mathsf{ck}, \mathsf{vk})$*, a signature* $\sigma$*, and randomization elements* $r_\Delta, u_\Delta$*, as input, outputs a new signature* $\sigma'$*.*

- $\mathsf{RS.Ver}(\mathsf{pk} = (\mathsf{pp}, \mathsf{vk}, \mathsf{ck}), \mathsf{cm}, \sigma) \to \{0, 1\}$*: is a deterministic algorithm, takes as input the public key* $\mathsf{pk}$*,* $\mathsf{cm} \in \mathcal{C}$ *and signature* $\sigma$*, outputs 1 for successful verification, otherwise 0.*

- $\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{pk} = (\mathsf{pp}, \mathsf{vk}, \mathsf{ck})) \to \{0, 1\}$ : *is a deterministic algorithm that takes in a secret key* $\mathsf{sk}$ *and verification key* $\mathsf{vk}$*, checks for consistency and returns 1 for success, 0 for failure*

**Definition 20 (RS Correctness).** *A rerandomizable signature scheme over commitments is correct if for all security parameters* $1^\lambda$*, for all* $\ell > 1$*, all bilinear groups* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \in [\mathsf{BGGen}(1^\lambda)]$*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \in [\mathsf{KeyGen}(\mathsf{BG}, 1^\ell)]$*, all messages* $m \in \mathcal{M}$*, all commitments* $\mathsf{cm} \in \mathcal{C}$*, all commitment keys* $\mathsf{ck} \in [\mathsf{CM.KeyGen}(1^\lambda)]$*, and all randomness* $r, u, r_\Delta, u_\Delta \in \mathbb{Z}_p$ *we have:*

$$\begin{aligned}
&\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{pk}) = 1 \quad \wedge \\
&\Pr\left[\mathsf{RS.Ver}(\mathsf{pk}, (\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{CM.Com}(\mathsf{ck}, m, r), u)), m) = 1\right] = 1 \quad \wedge \\
&\Pr\left[\mathsf{RS.Ver}(\mathsf{pk}, \mathsf{RS.Rerand}(\mathsf{pk}, (\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{CM.Com}(\mathsf{ck}, m, r), u)), r_\Delta, u_\Delta), m) = 1\right] = 1 \quad \wedge \\
&\Pr\left[\mathsf{RS.Ver}(\mathsf{pk}, (\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{CM.Com}(\mathsf{ck}, m, r + r_\Delta), u + u_\Delta)), m) = 1\right] = 1
\end{aligned}$$

**Definition 21 (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* $\mathsf{negl}(\lambda)$ *such that:*

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

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

Samneed to do this for the anonymous credential section

**Definition 22 (RS Signature Adaptation Under Malicious Keys).** *A rerandomizable signature scheme* $\mathsf{RS}$ *satisfies* signature adaptation under malicious keys *if for all tuples* $(\mathsf{pk}, \mathsf{cm}, \sigma, r)$ *where:*

- $\mathsf{cm} \in \mathcal{C}$ *is a valid commitment,*

- $\sigma$ *is a valid signature under* $\mathsf{pk}$ *(i.e.,* $\mathsf{RS.Ver}(\mathsf{pk}, \mathsf{cm}, \sigma) = 1$*),*

- $r \in \mathbb{Z}_p^*$*,*

*the distribution of* $\mathsf{RS.Rerand}(\mathsf{pk}, \sigma, \mu)$ *is identical to* $\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{CM.Rerand}(\mathsf{ck}, \mathsf{cm}, r))$*, even when* $\mathsf{pk}$ *is adversarially generated.*

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

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

$$\mathcal{R} \leftarrow \mathsf{PoK}\{(m_1, \ldots, m_\ell, u + u_\Delta, r + r_\Delta) :$$

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

- $\mathsf{RS.VerKey}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck}) \to \{0, 1\}$ : verifies the correctness of the issuers secret and verification key $(\mathsf{sk}, \mathsf{vk})$ and commitment key $\mathsf{ck}$:

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

**Security of our Construction**

**Theorem 2.** *RS is correct*

*Proof.* First we demonstrate the provers 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 the initial pairing.

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

Secondly, we need to verify knowledge of messages within the commitment. The Prover used $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$ during verification and this would be the natural method to 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 8, 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

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

Then runs a sigma protocol to prove

$$\mathsf{cm}' = g^{r \cdot r_\Delta} \prod_{i=1}^{\ell} g_i^{m_i}$$

**Theorem 3 (EUF-CMA Security).** *Assume the PS-LRSW assumption holds and the Pedersen commitment is computationally binding. Then, in the Algebraic Group Model, our rerandomizable signature scheme is existentially unforgeable under adaptive chosen-message(commitment) attacks. For any algebraic PPT adversary $\mathcal{A}$, there exist PPT reductions $\mathcal{B}_0, \mathcal{B}_1$ such that:*

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

*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 $\epsilon$.

*1. Setup* Given PS-LRSW challenge over bilinear groups $(g, \tilde{g}, X = g^x, \tilde{X} = \tilde{g}^x, Y = g^y, \tilde{Y} = \tilde{g}^y)$:

1. **Commitment Setup:** For 2-slot Pedersen:
   - Choose $\alpha_1, \alpha_2, \beta_1, \beta_2 \leftarrow\$ \mathbb{Z}_p$
   - Set $g_1 = Y^{\alpha_1} g^{\beta_1}$, $\tilde{g}_1 = \tilde{Y}^{\alpha_1} \tilde{g}^{\beta_1}$
   - Set $g_2 = Y^{\alpha_2} g^{\beta_2}$, $\tilde{g}_2 = \tilde{Y}^{\alpha_2} \tilde{g}^{\beta_2}$
2. **Public Key:** $\mathsf{pk} = (\tilde{X}, g_1, \tilde{g}_1, g_2, \tilde{g}_2)$
3. Send $\mathsf{pk}$ to $\mathcal{A}$. Distribution matches real scheme as $\alpha_i, \beta_i$ are random.

*2. Oracle Simulation* **Signing Oracle:** For query $(m_1, m_2, r)$:

- **Case $\mathcal{B}_0$ (PS Reduction):**
  1. Compute $m = \alpha_1 m_1 + \alpha_2 m_2$
  2. Query PS-LRSW oracle for $(h, h^{x+my})$
  3. Return $\sigma = (h, h^{x+my} \cdot h^{\beta_1 m_1 + \beta_2 m_2 + r})$
- **Case $\mathcal{B}_1$ (Binding Reduction):**
  1. Compute $\mathsf{cm} = g_1^{m_1} g_2^{m_2} g^r$
  2. Choose $u \leftarrow\$ \mathbb{Z}_p$, return $\sigma = (g^u, (X \cdot \mathsf{cm})^u)$

**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 Extraction* When $\mathcal{A}$ outputs forgery $(m_1^*, m_2^*, r^*, \sigma^* = (\sigma_1^*, \sigma_2^*))$:

**Case 1: New Message Combination** If $m^* = \alpha_1 m_1^* + \alpha_2 m_2^*$ is new:

– $\mathcal{B}_0$ computes:
$$(U, B/U^{\beta_1 m_1^* + \beta_2 m_2^* + r^*}) = (g^u, X^u Y^{m^* u})$$

– This breaks PS-LRSW as $m^*$ wasn't queried. Thus:
$$\epsilon_0 \geq \Pr[\text{New } m^*] - \frac{q_s}{p}$$

**Case 2: Commitment Collision** If $m^*$ exists in prior query $(m_1, m_2) \neq (m_1^*, m_2^*)$:

– $\mathcal{B}_1$ finds collision:
$$g_1^{m_1} g_2^{m_2} = g_1^{m_1^*} g_2^{m_2^*}$$

– Solving gives DLOG relation for $\alpha_i, \beta_i$, breaking binding:
$$\epsilon_1 \geq \Pr[\text{Collision}] - \frac{1}{p}$$

## 5 Multi Issuer Multi Credential Anonymous Credentials (MIMC-ABC)

See

Anonymous Credential Systems (ACS) [Cha85, CL04, ASM06, PS16, FHS19] address privacy concerns but face challenges balancing privacy with accountability, orthogonally, unconditionally anonymous payment systems have demonstrated how unconditional anonymity can enable system abuse. Current systems focus on protecting against sybil attacks [CKS24, RARM], enabling revocation [CL02, CDR16, CDH16, BCD+17], rich attribute-based credential authentication [RWGM22, BS23] but very few implement all. The tension between privacy and accountability has become increasingly critical as governments worldwide, particularly the EU's Digital Identity Framework [noa24], move toward privacy-preserving digital identity wallets. These challenges motivate the need for a comprehensive identity system that achieves privacy, accountability, and practical deployment requirements simultaneously.

### 5.1 Notation

The MIMC-ABC system is run for obtaining a master and context credential, differing in the obtain, issue and show, verify algorithms. We generalize the notation for elements but also subscript elements when expressed explicitly; $\mathsf{cm}$ is a commitment that could be Master or Context, whereas $\mathsf{cm}_\mathsf{m}, \mathsf{cm}_\mathsf{c}$ are explicitly for Master and Context, respectively.

We base our Multi Issuer, Multi Credential Multi-show Attribute based Anonymous Credentials off the model in [FHS19]

### 5.2 Syntax

**Definition 23 (MIMC-ABC System).** *A Multi-Issuer Multi-Credential Attribute-based Anonymous Credential system consists of the following* $\mathsf{PPT}$ *algorithms:*

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pp})$ *Takes security parameter* $\lambda$ *in unary, outputs public parameters* $\mathsf{pp}$.
- $\mathsf{OrgKeygen}(\mathsf{pp}, n) \rightarrow (\mathsf{osk}, \mathsf{opk})$*: is a probabilistic algorithm that takes public parameters* $\mathsf{pp}$ *and* $n$ *the upper bound of credential attributes. Outputs organisations keypair* $(\mathsf{osk}, \mathsf{opk})$
- $\mathsf{UserKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{usk})$*: is a probabilistic algorithm that takes public parameters* $\mathsf{pp}$*, outputs users secret key* $\mathsf{usk}$
- $(\mathsf{ObtainMaster}(\vec{m}, \mathsf{usk}, \mathsf{opk}, aux), \mathsf{IssueMaster}(\mathsf{osk}, \mathsf{cm}, aux)) \rightarrow (\mathsf{cred}_\mathsf{m}, \perp)$ *is an interactive protocol between a user and an issuing organization. The user inputs their message vector* $\vec{m}$ *and secret key* $\mathsf{usk}$. *The issuer inputs their secret key* $\mathsf{osk}$. *The protocol outputs a master credential* $\mathsf{cred}_\mathsf{m}$ *to the user and* $\perp$ *to the issuer.*
- $(\mathsf{ObtainContext}(\vec{m}, \mathsf{usk}, \mathsf{opk}, \mathsf{cred}_\mathsf{m}, aux), \mathsf{IssueContext}(\mathsf{osk}, \mathsf{cm}, \mathsf{opk}_m, aux)) \rightarrow (\mathsf{cred}_\mathsf{c}, \perp)$ *is an interactive protocol between a user and an issuing organization. The user inputs their message vector* $\vec{m}$*, secret key* $\mathsf{usk}$*, and master credential* $\mathsf{cred}_\mathsf{m}$. *The issuer inputs their secret key* $\mathsf{osk}$ *and the verification key of the master credential* $\mathsf{opk}_m$. *The protocol outputs a context credential* $\mathsf{cred}_\mathsf{c}$ *to the user and* $\perp$ *to the issuer.*
- $(\mathsf{Show}(\{\mathsf{cm}_i, r_i, \mathsf{cred}_i\}_{i=1}^n, \phi), \mathsf{Verify}(\{\mathsf{cred}'_i, \mathsf{opk}_i\}_{i=1}^n, \phi, \pi)) \rightarrow \{0, 1\}$ *is an interactive protocol between a user and verifier. The user runs* $\mathsf{Show}$ *with* $n$ *commitment-opening pairs* $(\mathsf{cm}_i, r_i)$*, their corresponding credentials* $\mathsf{cred}_i$*, and a predicate* $\phi$. *The verifier runs* $\mathsf{Verify}$ *with* $n$ *rerandomized credentials* $\mathsf{cred}'_i$*, their associated organization public keys* $\mathsf{opk}_i$*, the predicate* $\phi$*, and proof* $\pi$. *The protocol outputs 1 if verification succeeds, 0 otherwise.*

## 5.3 Construction

### Intuition of our Construction

**Outline** Our credential system operates over attribute space $\mathbb{Z}_p$ with arbitrary string attributes mapped to field elements by a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. A credential for user $i$ consists of a commitment $\mathsf{cm}_i \leftarrow \mathsf{CM.Commit}([m]; \mathsf{usk})$ where $m$ is e.g. $[\mathsf{id}, \mathsf{ctx}]$ with randomness denote by $\mathsf{usk} \leftarrow\!\!\$\ \mathbb{Z}_p^*$, a Pointcheval-Sanders signature $\sigma \leftarrow \mathsf{PS.Sign}(\mathsf{cm}_i, \mathsf{osk})$ on the commitment, verifiable by $\mathsf{PS.Verify}(\sigma, \mathsf{cm}_i, \mathsf{opk}) = 1$

**Freshness** To prevent replay attacks in credential show/verify protocols, we employ an interactive challenge-response mechanism as per Sigma protocols **Cite Sigma Protocols**. During showing, the verifier provides a random challenge that must be incorporated in the zero-knowledge proofs, ensuring uniqueness of each showing. Interaction could be removed using the Fiat-Shamir transform **Cite Fiat Shamir**, this would require verifiers to maintain a list of used proofs introducing overhead and potential security concerns in a multi-verifier/distributed setting.

**Malicious Organization Keys.** For our PS signature-based ABC system, we define an NP-relation $\mathcal{R}_O$ capturing well-formed organization keys:

$$((pk, pp_{cm}), (sk, r)) \in \mathcal{R}_O \iff \mathsf{PS.VKey}(sk, pk) = 1 \land pp_{cm} = \mathsf{Setup}(1^\lambda; r)$$

During credential issuance, the organization must provide a zero-knowledge proof of knowledge $\pi \leftarrow \mathsf{ZKPoK}\{(sk, r) : ((pk, pp_{cm}), (sk, r)) \in \mathcal{R}_O\}$. This ensures:

- The organization knows its PS signing key $sk$
- The commitment parameters $pp_{cm}$ are properly generated with known randomness $r$

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

## 5.4 Master Credential Protocol

The master credential issuance protocol enables a user to obtain their root credential, a rerandomizable signature over commitments from the Master Credential Issuer $\mathcal{I}_{\mathsf{m}}$. The Master Credential is a signature $\sigma_{\mathsf{m}}$ over commitment $\mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_m})$

$\underline{\mathsf{OrgKeyGen}(1^\lambda, 1^\ell)}$

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

$(\mathsf{sk_m}, \mathsf{vk_m}) \leftarrow\!\!\$\ \mathsf{RS.KeyGen}(\mathsf{ck_m}),\ \ \text{Return } (\mathsf{osk_m}, \mathsf{opk_m}) = ((\mathsf{sk_m}), (\mathsf{vk_m}, \mathsf{ck_m}))$

$\underline{\mathsf{UserKeyGen}(1^\lambda)} : \mathsf{usk} \leftarrow\!\!\$\ \mathbb{Z}_p,\ \text{Return } \mathsf{usk}$

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

$\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1) = \mathsf{ZKPoK}\{(\mathsf{cm}_1, \mathsf{usk}) | \mathsf{cm}_1 = g_1^0 g_2^0 g^{\mathsf{usk}}\}$

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

$\underline{\mathsf{Obtain}(\mathsf{usk_m}, \mathsf{opk_m})}$ $\qquad\qquad\qquad\qquad$ $\underline{\mathsf{Issue}(\mathsf{cm_m}, \tilde{\mathsf{m}}, \mathsf{osk_m})}$

If $\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck})$ fails, return $\perp$ $\quad \xleftarrow{\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk},\mathsf{vk},\mathsf{ck})}$

$\mathsf{cm}_1 = \mathsf{CM.Com}([0, 0]; \mathsf{usk})$ $\quad \xrightarrow{\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)}$ $\quad$ If $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)$ fails, return $\perp$

$\mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; 0),\ \mathsf{cm_m} = \mathsf{cm}_1 \cdot \mathsf{cm}_2$

If $\mathsf{cm_m} \neq \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk})$ or $\quad \xleftarrow{\mathsf{cm_m},\mathsf{id},\mathsf{ctx},\sigma_{\mathsf{m}}}$ $\quad$ $u \leftarrow\!\!\$\ \mathbb{Z}_p,\ \sigma_{\mathsf{m}} \leftarrow\!\!\$\ \mathsf{RS.Sign}(\mathsf{cm_m}, \mathsf{osk}, u)$

If $\mathsf{RS.Ver}(\sigma_{\mathsf{m}}, \mathsf{cm_m}, \mathsf{opk}) = 0$, return $\perp$

Else, return $\mathsf{cred_m} \leftarrow (\sigma_{\mathsf{m}}, \mathsf{cm_m}, \mathsf{opk_m})$

$\underline{(\mathsf{Show}, \mathsf{Verify}) :}$

$\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}') \mid \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk}') \wedge \mathsf{RS.Ver}(\sigma'_{\mathsf{m}}, \mathsf{opk_m}) = 1 \ \wedge\ \mathsf{ctx} = \text{"}\mathit{master}\text{"}\}$

$\underline{\mathsf{Show}(\mathsf{cred_m})}$ $\qquad\qquad\qquad\qquad$ $\underline{\mathsf{Verify}(\sigma'_{\mathsf{m}}, \mathsf{cm_m}', \mathsf{opk_m})}$

Let $\mathsf{cred_m} = (\sigma_{\mathsf{m}}, \mathsf{cm_m}, \mathsf{usk_m}, \mathsf{opk_m})$, sample $\mathsf{usk}_{\Delta}, u_{\Delta} \leftarrow\!\!\$\ \mathbb{Z}_p^2$

$\sigma'_{\mathsf{m}} = (\sigma_{\mathsf{m1}}^{u_\Delta}, (\sigma_{\mathsf{m2}} \cdot \sigma_{\mathsf{m1}}^{\mathsf{usk}_\Delta})^{u_\Delta}) \leftarrow \mathsf{RS.Rand}(\sigma_{\mathsf{m}}, \mathsf{usk}_\Delta, u_\Delta)$

$\mathsf{cm_m}' = (\mathsf{cm_m} \cdot g^{\mathsf{usk}_\Delta}) \leftarrow \mathsf{CM.Rand}(\mathsf{cm_m}, \mathsf{usk}_\Delta),\ \mathsf{usk}' = \mathsf{usk} + \mathsf{usk}_\Delta$

$\mathsf{cred_m}' = (\sigma'_{\mathsf{m}}, \mathsf{cm_m}', \mathsf{usk}', u_\Delta)$ $\quad \xrightarrow{\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}')}$ $\quad$ If $\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}')$ fails, return 0, else 1

Fig. 1: Master Credential Protocol

*Informal Security Analysis* The two-party process between the user and master credential oracle ensures sybil resistance of unique identifier $\mathsf{id}$; the oracle has access to the user information and checks duplicate issuance within their own identity system. $\mathcal{U}$'s $\mathsf{usk}$ remains hidden via the hiding property of the commitment $\mathsf{cm}_1$ malicious use prevented by $\Pi^{\mathcal{R}_{\mathsf{zero}}}$

## 5.5 Context Credential Protocol

The context credential issuance protocol enables a user to obtain a context credential linked to their root credential. The Context Credential $\mathsf{cred_c}$ contains a signature $\sigma_\mathsf{c}$ over commitment $\mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ issued by $\mathcal{I}_\mathsf{c}$

$\underline{(\mathsf{OrgKeyGen}, \mathsf{UserKeyGen}) :}$ proceed analogously to Master Credential

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

$\Pi^{\mathcal{R}_\mathsf{verkey}}$ proceed analogously to Master Credential

$\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}, \phi) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}) | \mathsf{cm_c} = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}} g^{\mathsf{usk_c}} \wedge \mathsf{ctx} = \text{"}\mathit{context}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{m}', \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{master}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Obtain}(\mathsf{usk_c}, \mathsf{opk_c}, \vec{m})}$ $\underline{\mathsf{Issue}(\mathsf{cm_c}, \mathsf{osk_c})}$

$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c})$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c})$ fails, return $\perp$.

If $\mathsf{cm_c} \neq \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ or $\xleftarrow{\mathsf{cm_c}, \mathsf{id}, \mathsf{ctx}, \sigma_\mathsf{c}}$ Else, $u \leftarrow\$ \mathbb{Z}_p$, $\sigma_\mathsf{c} \leftarrow\$ \mathsf{RS.Sign}(\mathsf{cm_c}, \mathsf{osk_c}, u)$

If $\mathsf{RS.Ver}(\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c}) = 0$, return $\perp$

Else, return $\mathsf{cred_c} \leftarrow (\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c})$

$\underline{(\mathsf{Show}, \mathsf{Verify}) :}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_m}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_m}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{m}', \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{master}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_c}') | \mathsf{cm_c}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{c}', \mathsf{opk_c}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{context}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Show}(\mathsf{cred_m})}$ $\underline{\mathsf{Verify}(\sigma_\mathsf{m}', \mathsf{cm_m}', \mathsf{opk_m})}$

Compute $\mathsf{cred_m}', \mathsf{cred_c}'$ analogously to Master Credential

$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}'), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}')$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')$ fails, return 0, Else 1

Fig. 2: Context Credential Protocol

**Informal Security Analysis** Sybil Resistance: The deterministic nullifier $\delta \leftarrow VRF(k, ctx)$ binds each context credential to a unique (user, context) pair, preventing multiple credentials for the same context. The reciprocal proof $\Pi_6$ ensures correct nullifier derivation from the master key $k$. Credential Binding: Context Credentials are bound to master credentials through shared identity $s$ and $\Pi_7$. The selective disclosure proof $\Pi_4$ ensures correct commitment structure without revealing private values. Privacy: The protocol only reveals $ctx$ and $attrs$ to CCO to allow identity verification while hiding $s$. The Master Credential $Cred_m$ remains unlinkable by being rerandomized and proven in zero knowledge it verifies with the Master Credential Oracles public key.

**Verification** A user $user$ wants to prove to any relying party $rely$ they have a valid credential that satisfies a verification statement $\phi$. The protocol takes as input $(rcd, ccd, \phi, rpk, acc, n)$ and outputs success or failure. $rely$ starts by sending $(\phi, n, acc)$ to $user$ where $\phi$ is a statement that specifies the requirements for a successful verification and $acc$ is the current accumulator value of revoked nullifiers. $user$ starts by randomizing their credentials $rcd' = psrerand(rcd)$ and $ccd' = psrerand(ccd)$ and verifies $psverify_{ck_{rcd}}(rcd')$ and $psverify_{ck_{ccd}}(ccd')$. $user$ generate their nullifiers $nullif_{pid} \leftarrow PRF_s(pid)$ and $nullif_{ctx} \leftarrow PRF_s(ctxid)$ and obtains non-membership witnesses $wpid$, $wctx$ for nullifiers against $acc$. $user$ generates a zero-knowledge proof $\pi$ showing 1) their credentials are valid, 2) they're not expired $(expiry > current_time)$, 3) their nullifiers are correctly formed from $s$, 4) their nullifiers are not in $acc$ using witnesses $wpid, wctx$, 5) the credential attributes satisfy $\phi$, 6) proof freshness using $n$. $user$ sends $(\pi, attrs_\phi)$ to $rely$, $rely$ verifies $\pi$ against $acc$ and validates $attrs_\phi$ satisfies $\phi$. Returns accept if all checks pass, reject otherwise.

## 6 Security Properties Framework

### 6.1 Base Properties (Single Credential)

- **Unforgeability:**
  - Type-1: Cannot forge new credentials (EUF-CMA)
  - Type-2: Cannot create false proofs about valid credentials (Soundness)
  - Note: Verification requires both signature validity AND proof soundness
- **Attribute Soundness:**
  - Committed attributes satisfy verification predicate
  - ZKP soundness ensures honest attribute revelation
  - Relationship to commitment scheme binding property
- **Base Anonymity:**
  - Unlinkable showings (rerandomization)
  - Selective disclosure via ZK proofs
  - User identity remains hidden

### 6.2 Extended Properties (Multi-Credential)

- **Cross-Credential Unforgeability:**
  - Extends Type-1: No forged credentials across contexts
  - Extends Type-2: No false cross-credential proofs
  - Implies: Base unforgeability in each context
- **Identity Consistency:**
  - Cross-credential identity binding
  - Attribute consistency across credentials
  - Requires: Knowledge of all credential openings
- **Multi-Show Anonymity:**
  - Cross-credential unlinkability while maintaining consistency
  - Private identity proofs across credentials
  - Implies: Base anonymity for each credential

Key changes between single and multiple issuer

1. Base Unforgeability + Identity Binding $\implies$ Cross-Credential Unforgeability
2. Base Anonymity + Consistent Identity $\implies$ Multi-Show Anonymity
3. Attribute Soundness + Cross-Credential Binding $\implies$ Identity Consistency

The reduction theorem flow should be 1. single credential security definitions 2. multi-credential multi-issuer security definitions 3. reduction theorem showing relationship (optional) 4. security proof leveraging reduction

## 7 Multi Issuer Multi Credential Anonymous Credentials (MIMC-ABC)

### 7.1 Notation

We base our Multi Issuer, Multi Credential Multi-show Attribute based Anonymous Credentials off the model in [FHS19] and extend it to support rerandomizable signatures over commitments and predicate-based zero-knowledge proof verification allowing users to prove statements about their committed and signed attributes without revealing any additional information.

### 7.2 Predicate Satisfaction

We define a predicate $\phi$ as a boolean function over a set of attributes $m$. Formally, $\phi : \mathcal{M} \to \{0,1\}$, where $\mathcal{M}$ is the message space. For a credential with attributes $m = [\mathsf{id}, \mathsf{ctx}, \ldots]$, we say that "$m$ satisfies $\phi$", denoted as $\phi(m) = 1$, if the boolean function evaluates to true on the attributes. **Example:** Consider a predicate $\phi_{\text{master}} = (\mathsf{ctx} = \texttt{"master"})$. An attribute set $m = [\mathsf{id} = 123, \mathsf{ctx} = \texttt{"master"}]$ satisfies $\phi_{\text{master}}$ because the ctx attribute equals "master". In the context of our unforgeability definition, we use predicate satisfaction to determine whether an adversary has produced a valid forgery or merely reused existing credentials in a legitimate way.

### 7.3 Syntax

**Definition 24 (MIMC-ABC System).** *A Multi-Issuer Multi-Credential Attribute-based Anonymous Credential system consists of the following* PPT *algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp})$ *Takes security parameter $\lambda$ in unary, outputs public parameters* pp.
- $\mathsf{OrgKeygen}(\mathsf{pp}, n) \to (\mathsf{osk}, \mathsf{opk})$*: Is a probabilistic algorithm that takes public parameters* pp *and $n$ the upper bound of credential attributes. Outputs organisation's keypair* $(\mathsf{osk}, \mathsf{opk})$
- $\mathsf{UserKeyGen}(\mathsf{pp}) \to (\mathsf{usk})$*: Is a probabilistic algorithm that takes public parameters* pp*, outputs user's secret key* usk
- $(\mathsf{ObtainMaster}(\vec{m}, \mathsf{usk_m}, \mathsf{opk_m}, \mathsf{aux}), \mathsf{IssueMaster}(\mathsf{osk_m}, \mathsf{cm_m}, \mathsf{aux})) \to (\mathsf{cred_m}, \perp)$ *is an interactive protocol between a user and an issuing organization. The user inputs their message vector $\vec{m} = [\mathsf{id}, \mathsf{ctx}, \ldots]$ containing a unique identifier* id *and context* ctx = *"master", along with secret key* $\mathsf{usk_m}$*. The issuer inputs their secret key* $\mathsf{osk_m}$*. The protocol outputs a master credential* $\mathsf{cred_m}$ *to the user and $\perp$ to the issuer.*
- $(\mathsf{ObtainContext}(\vec{m}, \mathsf{usk_c}, \mathsf{opk_m}, \mathsf{cred_m}, \mathsf{aux}), \mathsf{IssueContext}(\mathsf{osk_c}, \mathsf{cm_c}, \mathsf{opk}, \mathsf{aux})) \to (\mathsf{cred_c}, \perp)$ *is an interactive protocol between a user and an issuing organization. The user inputs their message vector $\vec{m} = [\mathsf{id}, \mathsf{ctx}, \ldots]$ with the same identifier* id *as in their master credential and context* ctx = *"context", along with secret key* usk *and master credential* $\mathsf{cred_m}$*. The issuer inputs their secret key* osk *and the verification key of the master credential issuer* $\mathsf{opk}_m$*. During the protocol, the user proves possession of a valid master credential with the same* id*. The protocol outputs a context credential* $\mathsf{cred_c}$ *to the user and $\perp$ to the issuer.*
- $(\mathsf{Show}(\{\mathsf{cred_m}, \mathsf{cred_c}\}, \{\mathsf{cm_m}, \mathsf{cm_c}\}, \{\mathsf{usk_m}, \mathsf{usk_c}\}, \phi), \mathsf{Verify}(\{\mathsf{cred_m}', \mathsf{cred_c}'\}, \{\mathsf{cm_m}', \mathsf{cm_c}'\}, \phi, \pi)) \to \{0,1\}$ *is an interactive protocol between a user and verifier. The user runs* Show *with their master and context credentials, corresponding commitments, openings, and a predicate $\phi$. The user generates a proof $\pi$ demonstrating: (1) both credentials are valid signatures over their respective commitments, (2) both commitments share the same identifier* id*, and (3) the attributes satisfy predicate $\phi$. The verifier runs* Verify *with the rerandomized credentials, commitments, the predicate $\phi$, and proof $\pi$. The protocol outputs 1 if verification succeeds, 0 otherwise.*

### 7.4 Construction

### 7.5 Master Credential Protocol

The master credential issuance protocol enables a user to obtain their root credential, a rerandomizable signature over commitments from the Master Credential Issuer $\mathcal{I}_\mathsf{m}$. The Master Credential is a signature $\sigma_\mathsf{m}$ over commitment $\mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_m})$

$\underline{\mathsf{OrgKeyGen}(1^\lambda, 1^\ell)}$

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

$(\mathsf{sk_m}, \mathsf{vk_m}) \leftarrow\!\$\ \mathsf{RS.KeyGen}(\mathsf{ck_m}),\ \text{ Return } (\mathsf{osk_m}, \mathsf{opk_m}) = ((\mathsf{sk_m}), (\mathsf{vk_m}, \mathsf{ck_m}))$

$\underline{\mathsf{UserKeyGen}(1^\lambda)} : \mathsf{usk} \leftarrow\!\$\ \mathbb{Z}_p,\ \text{ Return } \mathsf{usk}$

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

$\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1) = \mathsf{ZKPoK}\{(\mathsf{cm}_1, \mathsf{usk}) | \mathsf{cm}_1 = g_1^0 g_2^0 g^{\mathsf{usk}}\}$

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

$\underline{\mathsf{Obtain}(\mathsf{usk_m}, \mathsf{opk_m})}$ $\qquad\qquad$ $\underline{\mathsf{Issue}(\mathsf{cm_m}, \tilde{\mathsf{m}}, \mathsf{osk_m})}$

If $\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk}, \mathsf{vk}, \mathsf{ck})$ fails, return $\perp$ $\quad \xleftarrow{\Pi^{\mathcal{R}_{\mathsf{verkey}}}(\mathsf{sk},\mathsf{vk},\mathsf{ck})}$

$\mathsf{cm}_1 = \mathsf{CM.Com}([0, 0]; \mathsf{usk})$ $\quad \xrightarrow{\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)}$ $\quad$ If $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)$ fails, return $\perp$

$\mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; 0),\ \mathsf{cm_m} = \mathsf{cm}_1 \cdot \mathsf{cm}_2$

If $\mathsf{cm_m} \neq \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk})$ or $\quad \xleftarrow{\mathsf{cm_m}, \mathsf{id}, \mathsf{ctx}, \sigma_\mathsf{m}}$ $\quad$ $u \leftarrow\!\$\ \mathbb{Z}_p$, $\sigma_\mathsf{m} \leftarrow\!\$\ \mathsf{RS.Sign}(\mathsf{cm_m}, \mathsf{osk}, u)$

If $\mathsf{RS.Ver}(\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{opk}) = 0$, return $\perp$

Else, return $\mathsf{cred_m} \leftarrow (\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{opk_m})$

$\underline{(\mathsf{Show}, \mathsf{Verify}) :}$

$\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}') \mid \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk}') \wedge \mathsf{RS.Ver}(\sigma'_\mathsf{m}, \mathsf{opk_m}) = 1 \ \wedge\ \mathsf{ctx} = "master"\}$

$\underline{\mathsf{Show}(\mathsf{cred_m})}$ $\qquad\qquad$ $\underline{\mathsf{Verify}(\sigma'_\mathsf{m}, \mathsf{cm_m}', \mathsf{opk_m})}$

Let $\mathsf{cred_m} = (\sigma_\mathsf{m}, \mathsf{cm_m}, \mathsf{usk_m}, \mathsf{opk_m})$, sample $\mathsf{usk}_\Delta, u_\Delta \leftarrow\!\$\ \mathbb{Z}_p^2$

$\sigma'_\mathsf{m} = (\sigma_{\mathsf{m1}}^{u_\Delta}, (\sigma_{\mathsf{m2}} \cdot \sigma_{\mathsf{m1}}^{\mathsf{usk}_\Delta})^{u_\Delta}) \leftarrow \mathsf{RS.Rand}(\sigma_\mathsf{m}, \mathsf{usk}_\Delta, u_\Delta)$

$\mathsf{cm_m}' = (\mathsf{cm_m} \cdot g^{\mathsf{usk}_\Delta}) \leftarrow \mathsf{CM.Rand}(\mathsf{cm_m}, \mathsf{usk}_\Delta),\ \mathsf{usk}' = \mathsf{usk} + \mathsf{usk}_\Delta$

$\mathsf{cred_m}' = (\sigma'_\mathsf{m}, \mathsf{cm_m}', \mathsf{usk}', u_\Delta)$ $\quad \xrightarrow{\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}')}$ $\quad$ If $\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred_m}')$ fails, return 0, else 1

Fig. 3: Master Credential Protocol

*Informal Security Analysis* The two-party process between the user and master credential oracle ensures sybil resistance of unique identifier $\mathsf{id}$; the oracle has access to the user information and checks duplicate issuance within their own identity system. $\mathcal{U}$'s $\mathsf{usk}$ remains hidden via the hiding property of the commitment $\mathsf{cm}_1$ malicious use prevented by $\Pi^{\mathcal{R}_{\mathsf{zero}}}$

### 7.6 Context Credential Protocol

The context credential issuance protocol enables a user to obtain a context credential linked to their root credential. The Context Credential $\mathsf{cred_c}$ contains a signature $\sigma_\mathsf{c}$ over commitment $\mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ issued by $\mathcal{I}_\mathsf{c}$

$\underline{(\mathsf{OrgKeyGen}, \mathsf{UserKeyGen}) :}$ proceed analogously to Master Credential

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

$\Pi^{\mathcal{R}_\mathsf{verkey}}$ proceed analogously to Master Credential

$\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}, \phi) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}) | \mathsf{cm_c} = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}} g^{\mathsf{usk_c}} \wedge \mathsf{ctx} = \text{"}\mathit{context}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk}') \wedge \mathsf{RS.Ver}(\sigma'_\mathsf{m}, \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{master}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Obtain}(\mathsf{usk_c}, \mathsf{opk_c}, \vec{m})}$ $\underline{\mathsf{Issue}(\mathsf{cm_c}, \mathsf{osk_c})}$

$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c})$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c})$ fails, return $\perp$.

If $\mathsf{cm_c} \neq \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ or $\xleftarrow{\mathsf{cm_c}, \mathsf{id}, \mathsf{ctx}, \sigma_\mathsf{c}}$ Else, $u \leftarrow\!\!\$\ \mathbb{Z}_p$, $\sigma_\mathsf{c} \leftarrow\!\!\$\ \mathsf{RS.Sign}(\mathsf{cm_c}, \mathsf{osk_c}, u)$

If $\mathsf{RS.Ver}(\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c}) = 0$, return $\perp$

Else, return $\mathsf{cred_c} \leftarrow (\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c})$

$\underline{(\mathsf{Show}, \mathsf{Verify}) :}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_m}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_m}') \wedge \mathsf{RS.Ver}(\sigma'_\mathsf{m}, \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{master}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_c}') | \mathsf{cm_c}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c}') \wedge \mathsf{RS.Ver}(\sigma'_\mathsf{c}, \mathsf{opk_c}) = 1 \wedge \mathsf{ctx} = \text{"}\mathit{context}\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Show}(\mathsf{cred_m})}$ $\underline{\mathsf{Verify}(\sigma'_\mathsf{m}, \mathsf{cm_m}', \mathsf{opk_m})}$

Compute $\mathsf{cred_m}', \mathsf{cred_c}'$ analogously to Master Credential

$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}'), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}')$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')$ fails, return 0, Else 1

Fig. 4: Context Credential Protocol

## 8 Unforgeability: Single Issuer, Single Credential

Traditional unforgeability definitions are insufficient for anonymous credential systems because rerandomizable credentials and zero-knowledge proofs enable legitimate reuse that would be classified as forgeries under EUF-CMA security. Our definition addresses two critical attack vectors:

1. Creating new valid credentials not issued by legitimate authorities
2. Using legitimately obtained credentials to satisfy predicates their attributes do not support

*Example 1 (Credential Forgery):* An adversary produces a credential $\mathsf{cred} = (\sigma, \mathsf{cm} = \mathsf{CM.Com}([\text{age} = 17]; \mathsf{usk}))$ that passes signature verification $\mathsf{RS.Ver}(\sigma, \mathsf{cm}, \mathsf{vk}) = 1$ without being issued by the authority. This directly breaks the underlying signature scheme.

*Example 2 (Predicate Misuse):* A corrupt user with a valid credential for age = 17 attempts to satisfy $\phi = (\text{age} \geq 18)$ by generating an incorrect zero-knowledge proof. Our verification prevents attribute-predicate mismatches.

We formalize unforgeability through an experiment where an adversary interacts with credential oracles. The winning condition requires producing a credential showing that:

$$\mathsf{MIMC.Verify}(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*, \phi^*) = 1 \quad \wedge \quad \nexists (m, i) \in \mathsf{CRED} \text{ such that } i \in \mathsf{CU} \ \wedge \ \phi^*(m) = 1$$

This single-issuer, single-credential definition establishes the base case for our multi-issuer, multi-credential construction

**Notation and Key Concepts**

– $\mathsf{HU}$: Set of honest users whose secret keys are unknown to $\mathcal{A}$
– $\mathsf{CU}$: Set of corrupt users whose secret keys are known to $\mathcal{A}$
– $\mathsf{CRED}$: Set of tuples $(m, i, \mathsf{cm}, \sigma)$ where $m$ is the attribute vector, $i$ is the user index, $\mathsf{cm}$ is the commitment, and $\sigma$ is the signature
– $\phi$: Predicate function $\phi : \mathcal{M} \to \{0, 1\}$ defining attribute-based policy conditions
– $\mathsf{Show}(\sigma, \mathsf{cm}, \mathsf{usk}, \phi)$: Algorithm producing a showing $(\mathsf{cred}', \mathsf{cm}', \pi)$ for predicate $\phi$
– $\mathsf{Verify}(\mathsf{cred}', \mathsf{cm}', \pi, \phi)$: Algorithm verifying a credential showing against predicate $\phi$

```
Experiment $\mathrm{Exp}^{\mathsf{UNF}}_{\mathsf{SingleIssuer},\mathcal{A}}(\lambda)$

    // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
$\mathsf{HU} \leftarrow \emptyset,\ \mathsf{CU} \leftarrow \emptyset,\ \mathsf{CRED} \leftarrow \emptyset$

    // Adversary Interaction
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk}, \mathsf{ck})$
$\mathcal{A}$ outputs $(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*, \phi^*)$

    // Winning Condition
$\mathsf{MIMC.Verify}(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*, \phi^*) = 1 \wedge$
  $\nexists (m, i, \mathsf{cm}, \sigma) \in \mathsf{CRED}$ such that
  $i \in \mathsf{CU} \ \wedge\ \phi^*(m) = 1$
```

```
$\mathcal{O}_{\mathsf{HU}}(i)$
if $i \notin \mathsf{HU}$
    $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
    $\mathsf{usk}[i] \leftarrow\!\!\$\ \mathsf{UserKeyGen}(1^\lambda)$
return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
if $i \in \mathsf{HU} \wedge i \notin \mathsf{CU}$
    $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$
    return $\mathsf{usk}[i] \wedge$
    optionally return all $(m, i)$ for $i \in \mathsf{CRED}$

$\mathcal{O}_{\mathsf{ObtIss}}(i, m)$
if $i \notin \mathsf{HU}$, return $\perp$
    $\mathsf{cm} \leftarrow \mathsf{CM.Com}([m]; \mathsf{usk}[i])$
    $\sigma \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm})$
    $\mathsf{cred} \leftarrow (\sigma, \mathsf{cm})$
    $\mathsf{CRED} \leftarrow \mathsf{CRED} \cup (m, i, \mathsf{cm}, \sigma)$
return $\mathsf{cred}$

$\mathcal{O}_{\mathsf{Show}}(i, \phi)$
if $i \notin \mathsf{HU}$, return $\perp$
    Find all $(m, i, \mathsf{cm}, \sigma) \in \mathsf{CRED}$ where $\phi(m) = 1$
    if no such credential exists, return $\perp$
    else Select one such credential $(m, i, \mathsf{cm}, \sigma)$
        $(\mathsf{cred}', \mathsf{cm}', \pi) \leftarrow \mathsf{Show}(\sigma, \mathsf{cm}, \mathsf{usk}[i], \phi)$
return $(\mathsf{cred}', \mathsf{cm}', \pi)$
```

### 8.1 Security Reduction

**Theorem 4 (Single Issuer Unforgeability).** *If the rerandomizable signature scheme is* EUF-CMA-*secure, the commitment scheme is binding, and the zero-knowledge proof system is sound, then the single-issuer anonymous credential system satisfies unforgeability according to* $\mathrm{Exp}^{\mathsf{UNF}}_{\mathsf{SingleIssuer},\mathcal{A}}(\lambda)$.

*Proof.* We reduce security of the credential system to the EUF-CMA security of the underlying signature scheme. Given an adversary $\mathcal{A}$ that breaks credential unforgeability with non-negligible probability $\epsilon$, we build an adversary $\mathcal{B}$ that breaks EUF-CMA with related probability.

**Reduction Setup:** $\mathcal{B}$ receives verification key $\mathsf{vk}$ and signing oracle access $\mathcal{O}_{\mathsf{sign}}(\cdot)$ from the EUF-CMA challenger. It sets $\mathsf{opk} = \mathsf{vk}$ and simulates the credential system experiment.

**Oracle Simulation:**

- $\mathcal{O}_{\mathsf{HU}}$ and $\mathcal{O}_{\mathsf{CU}}$: Implemented per experiment definition, maintaining sets HU and CU.
- $\mathcal{O}_{\mathsf{ObtIss}}(i, m)$: For user $i$ and message $m$, $\mathcal{B}$ computes $\mathsf{cm} = \mathsf{CM.Com}(m; \mathsf{usk}_i)$, obtains $\sigma \leftarrow \mathcal{O}_{\mathsf{sign}}(\mathsf{cm})$, records $(m, i, \mathsf{cm}, \sigma)$ in CRED, and returns $(\sigma, \mathsf{cm})$.
- $\mathcal{O}_{\mathsf{Show}}(i, \phi)$: For honest user $i$ and predicate $\phi$, $\mathcal{B}$ finds credentials in CRED whose attributes satisfy $\phi$, generates a valid showing, and returns it.

**Forgery:** When $\mathcal{A}$ outputs forgery $(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*, \phi^*)$ satisfying:

$$\mathsf{MIMC.Verify}(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*, \phi^*) = 1 \quad \wedge \quad \nexists(m, i, \mathsf{cm}, \sigma) \in \mathsf{CRED} : i \in \mathsf{CU} \wedge \phi^*(m) = 1$$

By ZKP soundness, $\pi^*$ proves that:

$$\exists m^*, r^* \text{ such that } \mathsf{cm}'^* = \mathsf{CM.Com}(m^*; r^*) \quad \wedge \quad \mathsf{RS.Ver}(\mathsf{cred}'^*, \mathsf{cm}'^*, \mathsf{vk}) = 1 \quad \wedge \quad \phi^*(m^*) = 1$$

We claim $(\mathsf{cm}'^*, \mathsf{cred}'^*)$ is a valid $\mathsf{EUF\text{-}CMA}$ forgery:

1. **Case 1:** If $\mathsf{cm}'^*$ is not a rerandomization of any commitment in $\mathsf{CRED}$, then $\mathsf{cm}'^*$ was never queried to $\mathcal{O}_{\mathsf{sign}}$, making it a valid $\mathsf{EUF\text{-}CMA}$ forgery.

2. **Case 2:** If $\mathsf{cm}'^*$ is a rerandomization of some $\mathsf{cm}_j \in \mathsf{CRED}$, then by commitment binding property, $m^* = m_j$. Two subcases arise:

   – If user $j \in \mathsf{HU}$, then $\mathcal{A}$ cannot know $\mathsf{usk}_j$ to generate a valid rerandomization.

   – If user $j \in \mathsf{CU}$, then by our winning condition, $\phi^*(m_j) = 0$, contradicting the proof assertion that $\phi^*(m^*) = 1$ given $m^* = m_j$.

Therefore, $\mathcal{B}$ outputs $(\mathsf{cm}'^*, \mathsf{cred}'^*)$ as its $\mathsf{EUF\text{-}CMA}$ forgery with success probability at least $\epsilon$.

## 8.2 Progression to Multi-Credential Unforgeability

Progression from a single-credential to multiple-credential system where a user holds multiple credentials and uses them together to satisfy complex predicate based authentication introduces new attack vectors which our enhanced security model should address. New attack vectors include:

1. **Credential Attribute Manipulation**: Modifying attributes within otherwise legitimate credentials

2. **Credential Binding Attacks**: Breaking the binding between credentials that should share the same identity

3. **Mix-and-Match Attacks**: Combining credentials from different identities to create unauthorized credential combinations

*Example 3 (Credential Binding Attack)* A corrupt user possesses legitimate credentials $\mathsf{cred}_{\mathsf{m}A}$ with $\mathsf{id} = 123$ and $\mathsf{cred}_{\mathsf{c}B}$ with $\mathsf{id} = 456$. They attempt to present these together with a forged zero-knowledge proof $\pi^*$ falsely claiming that both credentials share the same identity.

*Example 4 (Mixed Predicate Misuse)* A corrupt user with a master credential claiming age = 25 and a context credential claiming drivingClass = "motorcycle" attempts to satisfy the predicate $\phi = (\text{age} \geq 21 \wedge \text{drivingClass} = \text{"car"})$ by forging an incorrect zero-knowledge proof.

**Adversary Winning Condition** We formalize multi-credential unforgeability through an experiment where an adversary interacts with credential oracles and outputs a credential showing. The winning condition requires:

$$\begin{align}
&\mathsf{MIMC.Verify}(\mathsf{cred_m}'^*, \mathsf{cred_c}'^*, \mathsf{cm_m}'^*, \mathsf{cm_c}'^*, \pi^*, \phi^*) = 1 \quad \wedge \tag{1}\\
&\nexists\, i \in \mathsf{CU} \text{ such that} \tag{2}\\
&(id_i, \mathsf{attrs_m}, i, \cdot, \cdot) \in \mathsf{CRED}_M \quad \wedge \tag{3}\\
&(id_i, \mathsf{attrs_c}, i, \cdot, \cdot) \in \mathsf{CRED}_C \quad \wedge \tag{4}\\
&\phi^*((id_i, \mathsf{attrs_m}), (id_i, \mathsf{attrs_c})) = 1 \tag{5}\\
& \tag{6}
\end{align}$$

Informally, the adversary wins if they can produce credentials that:

1. Pass the verification process (signatures validate, zero-knowledge proofs verify)
2. **AND** no corrupt user possesses both a master and context credential with:
   - The same identity across both credentials
   - Attributes that would legitimately satisfy the chosen predicate

This definition captures our security goal: preventing adversaries from creating credential showings they shouldn't be able to produce based on the credentials legitimately issued to corrupt users.

## 8.3 Unforgeability: Single-Issuer, Multi-Credential

We now extend our unforgeability definition to address the case where a single issuer provides multiple credential types that must be used together in credential presentations.

```
Experiment $\mathrm{Exp}^{\mathsf{UNF}}_{\mathsf{SingleIssuer\text{-}MultiCred},\mathcal{A}}(\lambda)$
    // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$    // 1 issuer
$\mathsf{HU} \leftarrow \emptyset,\ \mathsf{CU} \leftarrow \emptyset$
$\mathsf{CRED}_M \leftarrow \emptyset, \mathsf{CRED}_C \leftarrow \emptyset$
    // CREDi: (id, attrs, user, commitment, signature)

    // Adversary Interaction
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}},\mathcal{O}_{\mathsf{CU}},\mathcal{O}_{\mathsf{ObtainM}},\mathcal{O}_{\mathsf{ObtainC}},\mathcal{O}_{\mathsf{Show}}}(\mathsf{opk}, \mathsf{ck})$
$\mathcal{A}$ outputs $(\mathsf{cred_m}'^*, \mathsf{cred_c}'^*, \mathsf{cm_m}'^*, \mathsf{cm_c}'^*, \pi^*, \phi^*)$

    // Winning Condition
$\mathsf{MIMC.Verify}(\mathsf{cred_m}'^*, \mathsf{cred_c}'^*, \mathsf{cm_m}'^*, \mathsf{cm_c}'^*, \pi^*, \phi^*) = 1 \wedge$
    $\nexists i \in \mathsf{CU}$ such that:
        $(\mathsf{id}_i, \mathsf{attrs_m}, i, \cdot, \cdot) \in \mathsf{CRED}_M \wedge$
        $(\mathsf{id}_i, \mathsf{attrs_c}, i, \cdot, \cdot) \in \mathsf{CRED}_C \wedge$
        $\phi^*((id_i, \mathsf{attrs_m}), (id_i, \mathsf{attrs_c})) = 1$
```

```
$\mathcal{O}_{\mathsf{ObtainM}}(i, \mathsf{attrs_m})$
if $i \notin \mathsf{HU}$, return $\perp$
    $\mathsf{cm_m} \leftarrow \mathsf{CM.Com}([id_i, \mathsf{attrs_m}]; \mathsf{usk}[i])$
    $\sigma_\mathsf{m} \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm_m})$
    $\mathsf{cred_m} \leftarrow (\sigma_\mathsf{m}, \mathsf{cm_m})$
    $\mathsf{CRED}_M \leftarrow \mathsf{CRED}_M \cup (id_i, \mathsf{attrs_m}, i, \mathsf{cm_m}, \sigma_\mathsf{m})$
return $\mathsf{cred_m}$
```

```
$\mathcal{O}_{\mathsf{ObtainC}}(i, \mathsf{attrs_c}, \mathsf{cred_m})$
if $i \notin \mathsf{HU}$, return $\perp$
    // Verify $\mathsf{cred_m}$
if $\nexists(\mathsf{id}_i, \mathsf{attrs_m}, i, \mathsf{cm_m}, \sigma_\mathsf{m}) \in \mathsf{CRED}_M$
    with $\mathsf{cred_m} = (\sigma_\mathsf{m}, \mathsf{cm_m})$, return $\perp$
        // Create $\mathsf{cred_c}$ bound to $\mathsf{id}$
    $\mathsf{cm_c} \leftarrow \mathsf{CM.Com}([\mathsf{id}_i, \mathsf{attrs_c}]; \mathsf{usk}[i])$
        // Simulate $\Pi^{\mathcal{R}_{\mathsf{eq}}}(\mathsf{cm_m}, \mathsf{cm_c})$
    $\Pi^{\mathcal{R}_{\mathsf{eq}}} \leftarrow \mathsf{ZK.Prove}(\mathsf{cm_m}, \mathsf{cm_c}, \mathsf{id}_i, \phi_{\mathsf{eq}})$
    $\sigma_\mathsf{c} \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm_c}, \mathsf{cm_m}, \pi_{eq})$
    $\mathsf{cred_c} \leftarrow (\sigma_\mathsf{c}, \mathsf{cm_c})$
    $\mathsf{CRED}_C \leftarrow \mathsf{CRED}_C \cup (id_i, \mathsf{attrs_c}, i, \mathsf{cm_c}, \sigma_\mathsf{c})$
return $\mathsf{cred_c}$
```

```
$\mathcal{O}_{\mathsf{Show}}(i, \phi)$
if $i \notin \mathsf{HU}$, return $\perp$
    Find $(id_i, \mathsf{attrs_m}, i, \mathsf{cm_m}, \sigma_\mathsf{m}) \in \mathsf{CRED}_M$ and
    $(id_i, \mathsf{attrs_c}, i, \mathsf{cm_c}, \sigma_\mathsf{c}) \in \mathsf{CRED}_C$ such that
    $\phi((id_i, \mathsf{attrs_m}), (id_i, \mathsf{attrs_c})) = 1$
    if no such credential pair exists, return $\perp$
        $(\mathsf{cred_m}', \mathsf{cred_c}', \mathsf{cm_m}', \mathsf{cm_c}', \pi) \leftarrow$
$\mathsf{Show}(\sigma_\mathsf{m}, \mathsf{cm_m}, \sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{usk}[i], \phi)$
return $(\mathsf{cred_m}', \mathsf{cred_c}', \mathsf{cm_m}', \mathsf{cm_c}', \pi)$
```

## 8.4 Completing the Single-Issuer, Multi-Credential Unforgeability Reduction

**1. Formalized Oracle Specifications** First, let's precisely define your oracles with explicit tracking of credential relationships:

```
Setup and KeyGeneration
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
$\mathsf{HU} \leftarrow \emptyset,\ \mathsf{CU} \leftarrow \emptyset$
$\mathsf{CRED}_M, \mathsf{CRED}_C \leftarrow \emptyset, \emptyset$
$\mathsf{OWNR} \leftarrow \emptyset$   // Maps credential indices to user identities


$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathbf{if}\ i \in \mathsf{HU} \cup \mathsf{CU}, \mathbf{return}\ \perp$
$\mathsf{usk}[i] \leftarrow\!\!\$\ \mathsf{UserKeyGen}(1^\lambda)$
$\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
$\mathbf{return}\ \mathsf{opk}$


$\mathcal{O}_{\mathsf{CU}}(i)$
$\mathbf{if}\ i \notin \mathsf{HU}, \mathbf{return}\ \perp$
$\mathsf{HU} \leftarrow \mathsf{HU} \setminus \{i\}$
$\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$
$\mathbf{return}\ \mathsf{usk}[i]$ and all $\mathrm{CRED}_j$ where $\mathsf{OWNR}[j] = i$


$\mathcal{O}_{\mathsf{ObtainM}}(i, \mathsf{attrs_m})$
$\mathbf{if}\ i \notin \mathsf{HU}, \mathbf{return}\ \perp$
$\mathsf{id}_i \leftarrow\!\!\$\ \{0,1\}^\lambda$   // Generate unique identifier
$\mathsf{cm_m} \leftarrow \mathsf{CM.Com}([\mathsf{id}_i, \mathsf{attrs_m}]; \mathsf{usk}[i])$
$\sigma_\mathsf{m} \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm_m})$
$\mathsf{cred_m} \leftarrow (\sigma_\mathsf{m}, \mathsf{cm_m})$
$j \leftarrow |\mathsf{CRED}_M| + 1$   // New credential index
$\mathsf{CRED}_M \leftarrow \mathsf{CRED}_M \cup \{j\}$
$\mathsf{OWNR}[j] \leftarrow i$
$_M[j] \leftarrow (\mathsf{id}_i, \mathsf{attrs_m})$
$\mathbf{return}\ \mathsf{cred_m}$


$\mathcal{O}_{\mathsf{ObtainC}}(i, \mathsf{attrs_c}, j_m)$
$\mathbf{if}\ i \notin \mathsf{HU}, \mathbf{return}\ \perp$
$\mathbf{if}\ j_m \notin \mathsf{CRED}_M \vee \mathsf{OWNR}[j_m] \neq i, \mathbf{return}\ \perp$   // Verify master credential ownership
$(\mathsf{id}_i, \mathsf{attrs_m}) \leftarrow_M [j_m]$   // Extract identity from master credential
$\mathsf{cm_c} \leftarrow \mathsf{CM.Com}([\mathsf{id}_i, \mathsf{attrs_c}]; \mathsf{usk}[i])$
  // Formalized binding proof verification
$\pi_{eq} \leftarrow (\mathsf{cm_m}_{j_m}, \mathsf{cm_c}, \mathsf{id}_i, \mathsf{usk}[i])$
$\mathbf{if}\ (\pi_{eq}, \mathsf{cm_m}_{j_m}, \mathsf{cm_c}) = 0, \mathbf{return}\ \perp$
$\sigma_\mathsf{c} \leftarrow \mathsf{Issue}(\mathsf{osk}, \mathsf{cm_c})$
$\mathsf{cred_c} \leftarrow (\sigma_\mathsf{c}, \mathsf{cm_c})$
$j_c \leftarrow |\mathsf{CRED}_C| + 1$
$\mathsf{CRED}_C \leftarrow \mathsf{CRED}_C \cup \{j_c\}$
$\mathsf{OWNR}[j_c] \leftarrow i$
$_C[j_c] \leftarrow (\mathsf{id}_i, \mathsf{attrs_c})$
$\mathsf{LINK}[j_c] \leftarrow j_m$   // Track which master credential authorized this context credential
$\mathbf{return}\ \mathsf{cred_c}$
```

```
MIMC.Verify($\mathsf{cred_m}', \mathsf{cred_c}', \mathsf{cm_m}', \mathsf{cm_c}', \pi, \phi$)
    // Signature verification on both credentials
if $\mathsf{RS.Ver}(\mathsf{cred_m}', \mathsf{cm_m}', \mathsf{opk}) = 0$, return $0$
if $\mathsf{RS.Ver}(\mathsf{cred_c}', \mathsf{cm_c}', \mathsf{opk}) = 0$, return $0$
    // Verify ZK proof demonstrating:
    // 1. Both commitments are well-formed
    // 2. Both commitments share the same identity
    // 3. Committed attributes satisfy predicate
if $(\pi, \mathsf{cm_m}', \mathsf{cm_c}', \phi) = 0$, return $0$
return $1$
```

```
$(\pi, \mathsf{cm_m}', \mathsf{cm_c}', \phi)$
    // Verify that $\pi$ proves the following statement:
$\exists \mathsf{id}, \mathsf{attrs_m}, \mathsf{attrs_c}, r_m, r_c$ such that:
$\mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{attrs_m}]; r_m) \wedge$
$\mathsf{cm_c}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{attrs_c}]; r_c) \wedge$
$\phi((\mathsf{id}, \mathsf{attrs_m}), (\mathsf{id}, \mathsf{attrs_c})) = 1$
```

**3. Complete Reduction Framework**

*Proof.* We construct a reduction from the unforgeability of our multi-credential system to the $\mathsf{EUF\text{-}CMA}$ security of the underlying signature scheme. Given an adversary $\mathcal{A}$ breaking our definition with probability $\epsilon$, we build an adversary $\mathcal{B}$ breaking $\mathsf{EUF\text{-}CMA}$.

**Setup:** $\mathcal{B}$ receives $\mathsf{vk}$ and oracle access to $\mathcal{O}_{\mathsf{sign}}(\cdot)$ from the $\mathsf{EUF\text{-}CMA}$ challenger. It sets $\mathsf{opk} = \mathsf{vk}$ and simulates our experiment.

**Oracle Simulation:** $\mathcal{B}$ implements $\mathcal{O}_{\mathsf{HU}}$, $\mathcal{O}_{\mathsf{CU}}$ as described. For $\mathcal{O}_{\mathsf{ObtainM}}$ and $\mathcal{O}_{\mathsf{ObtainC}}$, $\mathcal{B}$ uses $\mathcal{O}_{\mathsf{sign}}(\cdot)$ to obtain signatures, carefully tracking all credential data in $\mathsf{CRED_m}$, $\mathsf{CRED_c}$, $\mathsf{OWNR}$, $\mathsf{ATTR_m}$, $\mathsf{ATTR_c}$, and $\mathsf{LINK}$.

**Forgery Extraction:** When $\mathcal{A}$ outputs $(\mathsf{cred_m}'^*, \mathsf{cred_c}'^*, \mathsf{cm_m}'^*, \mathsf{cm_c}'^*, \pi^*, \phi^*)$ satisfying our winning condition, $\mathcal{B}$ proceeds with case analysis:

**Case 1:** If either $\mathsf{cm_m}'^*$ or $\mathsf{cm_c}'^*$ is not a rerandomization of any commitment in our records, $\mathcal{B}$ returns the corresponding signature as an $\mathsf{EUF\text{-}CMA}$ forgery.

**Case 2:** If both credentials are rerandomizations of existing commitments but from different users:

– By the binding property of commitments and soundness of the ZK system, the proof $\pi^*$ must demonstrate that both commitments share the same identity $\mathsf{id}^*$.

– However, the original commitments contained different identities, creating a contradiction.

– Therefore, the adversary must have broken either commitment binding or ZK soundness to produce $\pi^*$.

**Case 3:** If both credentials are rerandomizations from the same corrupt user but don't satisfy $\phi^*$:

– The ZK proof incorrectly claims that attributes satisfy $\phi^*$, contradicting soundness.

In all cases, we either extract a direct $\mathsf{EUF\text{-}CMA}$ forgery or establish a contradiction with a fundamental security property, completing our reduction.

### 8.5 Progression to Multi-Issuer, Multi-Credential Unforgeability

## 9 Sybil Resistance via VRF with Committed Attributes

Anonymous Credential systems have the paradoxical problem of requiring user privacy but preventing sybil attacks where users create multiple credentials. Traditional Sybil resistance mechanisms often compromise user anonymity by revealing identifying information or relationships between credentials. Verifiable Random Functions (VRFs) offer a promising solution by enabling the generation of verifiably pseudorandom and deterministic nullifiers from user-specific information, suitable for presentation to an issuer or for revocation lists. Existing VRF-based schemes often rely on computationally intensive bilinear pairings or reveal user attributes, introducing overhead or privacy risks.

Sybil Resistance - Single-credential: UnlinkOne identity per context - Multi-credential: Cross-context consistency - - Cannot obtain multiple credentials for same (identity, context) pair - - Master credential properly controls derivation of context credentials

We improve the state of the art by creating a lightweight VRF construction tailored for Anonymous Credential systems with 3 contributions:

1. **Pairing-Free VRF in Prime-Order Groups:** We adapt the Dodis-Yampolskiy VRF structure to function efficiently in standard prime-order groups, achieving provable pseudorandomness under the $q$-Diffie-Hellman Inversion ($q$-DHI) assumption.

2. **Zero-Knowledge Proof of Multiplicative Inverse:** We introduce a novel $\Sigma$-protocol that proves the multiplicative inverse relation between committed values $m_1 = k + \mathsf{ctx}$ and $m_2 = 1/m_1$, we use it in our scheme to verify the correctness of the VRF nullifier without revealing user secrets. We show it generalizes naturally for similar requirements in $\Sigma$-protocols, especially those needing to prove the q-DHI.

3. **Formal Security Guarantees:** We demonstrate sybil resistance reduces to the security of our construction, the unique provability of the vrf and the soundness of our $\Sigma$-protocol.

We first present the preliminaries and foundations of our VRF for committed inputs, the design of our $\Sigma$-protocol, and demonstrate how the integration achieves sybil resistance in anonymous credential systems.

**Definition 25 (Sybil-Resistant Issuance).** *For any context* $\mathsf{ctx}$*, a Context Credential issuer must be able to detect if a user with master credential containing identifier* $\mathsf{id}$ *has previously obtained a context credential for* $\mathsf{ctx}$*, without learning* $\mathsf{id}$ *itself or linking this issuance request to other credential presentations.*

### 9.1 Problem

To understand how we use the VRF within our application, we introduce the application: Within our anonymous credential system, a user has a Master Credential with a VRF key $k$ and Context Credential with $\mathsf{ctx}$, where $\mathsf{ctx}$ is the context, such as $\mathcal{H}(\textit{"DriversLicense"})$ where $\mathcal{H}$ is hashes a string to $\mathbb{Z}_p$

$$\mathsf{cm_m} = \mathsf{CM.Com}([k, \ldots]; r) = g_1^k \ldots h^r \quad \wedge \quad \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{ctx}, \ldots]; r_2) = g_1^{\mathsf{ctx}} \ldots h^{r_2}$$

During Context Credential issuance, a user must prove to the issuer that their context credential hasn't been issued before, that is, the Context Credential issuance must be *Sybil Resistant.* Our goal is to generate a unique, unlinkable nullifier for a specific context containing something in both the Master Credential and the Context Credential to protect the system from Sybil attacks while also retaining user privacy.

We leverage the structure and properties of the Dodis Yampolisky Verifiable Random Function (VRF)

$$\text{Nullifier } \mathsf{N} = g^{1/k+\mathsf{ctx}}$$

The Nullifier takes on the properties of correctness, pseudorandomness, and provable uniqueness from the VRF which we exploit in our protocol.

### 9.2 Preliminaries

**Definition 26 (q-DHI Assumption).** *Let $\mathbb{G}$ be a cyclic group of prime order $p$ with generator $g$. The q-Diffie-Hellman Inversion (q-DHI) assumption [MSK02] states that for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that:*

$$\Pr\left[x \leftarrow\!\!\$\ \mathbb{Z}_p^*, \quad \mathcal{A}(g, g^x, g^{x^2}, \ldots, g^{x^q}) = g^{1/x}\right] \leq \mathsf{negl}(\lambda)$$

*where the probability is taken over the random choice of $x$ and the random coins of $\mathcal{A}$. Informally, no* PPT *adversary can distinguish between $g^{1/\alpha}$ from a random group element.*

*Remark 5.* The $q$-DHI assumption is equivalent to the $(q+1)$-generalized Diffie-Hellman assumption (GDH) as shown by Boneh and Boyen [BB04]. This equivalence provides a solid theoretical foundation for our VRF construction's security.

**Definition 27 (Verifiable Random Function in Prime-Order Group).** *A Verifiable Random Function (VRF) in prime-order group $\mathbb{G}$ of order $q$ is a tuple of PPT algorithms* $(\mathsf{VRF.Gen}, \mathsf{VRF.Eval}, \mathsf{VRF.Vfy})$ *with associated message space $\mathcal{X}$, output space $\mathcal{N}$, and proof space $\Pi$, defined as:*

- $\mathsf{VRF.Gen}(1^\lambda) \to (sk, pk)$ : *Samples secret key $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$, computes public key $pk \leftarrow g^\alpha$, returns $(sk = \alpha, pk)$*
- $\mathsf{VRF.Eval}(sk, x) \to \mathsf{N}, \pi$ : *Returns output $\mathsf{N} \leftarrow g^{1/(x+sk)}$ and $\pi$ verifies the output $\mathsf{N}$*
- $\mathsf{VRF.Vfy}(pk, x, \mathsf{N}, \pi) \to \{0,1\}$ : *validates proof $\pi$ that $\mathsf{N} = g^{1/(x+sk)}$, outputs 1 for success, 0 for failure*
- **Correctness:** For all $(sk, pk) \leftarrow \mathsf{VRF.Gen}(1^\lambda)$ and all $x \in \mathcal{X}$:

$$\Pr\begin{bmatrix} (y, \pi) \leftarrow \mathsf{VRF.Eval}(sk, x) \\ 1 \leftarrow \mathsf{VRF.Vfy}(pk, x, \mathsf{N}, \pi) \end{bmatrix} = 1$$

- **Unique Provability:** For any $pk$ (possibly malicious) and $x \in \mathcal{X}$, no PPT adversary $\mathcal{A}$ can find two distinct pairs of outputs $(\mathsf{N}_0, \pi_0) \neq (\mathsf{N}_1, \pi_1)$ such that:

$$\mathsf{VRF.Vfy}(pk, x, \mathsf{N}_0, \pi_0) = \mathsf{VRF.Vfy}(pk, x, \mathsf{N}_1, \pi_1) = 1$$

- **Pseudorandomness:** For every PPT adversary , there exists negligible function $\mathsf{negl}(\lambda)$ such that:

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{PR}}_{\mathsf{VRF}}(\mathcal{A}, \lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

  where the pseudorandomness experiment $\mathsf{Exp}^{\mathsf{PR}}_{\mathsf{VRF}}$ is defined in the standard framework for VRFs.

### 9.3 Algebraic Analysis of Dodis Yampolskiy VRF

We first recall the classical Dodis Yampolskiy VRF construction with bilinear pairings, we demonstrate with Type-3 pairings as they are generally used in practice. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be groups of a bilinear map with prime order $p$ where $g_1, g_2$ are generators for $\mathbb{G}_1, \mathbb{G}_2$ respectively and $e$ is an efficient map from $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$:

- $\mathsf{VRF.Gen}(1^\lambda)$: Samples $k \leftarrow\!\!\$\ \mathbb{Z}_p$, set $pk = g^k$
- $\mathsf{VRF.Eval}(k, \mathsf{ctx}) \to (\mathrm{N}, \pi)$: $\pi = e(g_1, g_2)^{1/(k+\mathsf{ctx})}$, $\mathsf{N} = g_2^{1/(k+\mathsf{ctx})}$
- $\mathsf{VRF.Vfy}(pk, \mathsf{ctx}, \mathsf{N}, \pi) \to \{0,1\}$: assert $\quad e(g^{\mathsf{ctx}} \cdot pk, \mathsf{N}) \stackrel{?}{=} e(g_1, g_2) \quad \wedge \quad \pi \stackrel{?}{=} e(g_1, \mathsf{N})$

$\mathsf{Eval}$ computes the nullifier $\mathsf{N}$ and generates a proof $\pi$ to prove that anyone in possession of $pk$ and the input $\mathsf{ctx}$ can verify $\mathsf{N}$ was computed correctly. $\mathsf{Vfy}$ resembles a signature verification as it binds the public key $pk$, input $\mathsf{ctx}$, and nullifier output together.

The first pairing binds the public input $pk, \mathsf{ctx}$ with $\mathsf{N}$

$$\begin{aligned}
e(g_1^{\mathsf{ctx}} \cdot pk, \mathsf{N}) &\stackrel{?}{=} e(g_1, g_2) \\
e(g_1^{\mathsf{ctx}}, g_2^{1/(k+\mathsf{ctx})}) \cdot e(pk, g_2^{1/(k+\mathsf{ctx})}) &\stackrel{?}{=} e(g_1, g_2) \\
e(g_1^{\mathsf{ctx}}, g_2)^{1/(k+\mathsf{ctx})} \cdot e(g_1^k, g_2)^{1/(k+\mathsf{ctx})} &\stackrel{?}{=} e(g_1, g_2) \\
e(g_1, g_2)^{\mathsf{ctx}/(k+\mathsf{ctx})} \cdot e(g_1^k, g_2)^{k/(k+\mathsf{ctx})} &\stackrel{?}{=} e(g_1, g_2) \\
e(g_1, g_2)^{\mathsf{ctx}+k/(k+\mathsf{ctx})} &= e(g_1, g_2)
\end{aligned}$$

$$\begin{aligned}
\pi &\stackrel{?}{=} e(g_1, \mathsf{N}) \\
e(g_1, g_2)^{1/(k+\mathsf{ctx})} &\stackrel{?}{=} e(g_1, g_2^{1/(k+\mathsf{ctx})}) \\
e(g_1, g_2)^{1/(k+\mathsf{ctx})} &\stackrel{?}{=} e(g_1, g_2)^{1/(k+\mathsf{ctx})}
\end{aligned}$$

And the second pairing binds the proof $\pi$ to N

**Informal Security analysis of Bilinear Pairing VRF**

- **Correctness:** Follows directly from pairing properties. The algebraic structure ensures verification equations hold when computed honestly.
- **Unique Provability:** Each nullifier $\mathsf{N} = g_2^{1/(k+\mathsf{ctx})}$ is uniquely determined by the pairing equation $e(g_1^{k+\mathsf{ctx}}, \mathsf{N}) = e(g_1, g_2)$. A forgery requires solving DLOG to create $g_2^{1/(k+\mathsf{ctx}')} = \mathsf{N}$.
- **Pseudorandomness:** Relies on the $q$-DHI assumption in bilinear groups. Given $g_1, g_1^k, g_1^{k^2}, \ldots$, distinguishing $\mathsf{N} = g_2^{1/(k+\mathsf{ctx})}$ from random reduces to computing $g_2^{1/k}$ (a $q$-DHI instance).

## 9.4 VRF with Committed Inputs

As demonstrated above, the classical Dodis-Yampolskiy VRF uses pairings to verify the relationship between inputs and outputs through the equation: $e(g_1^{\mathsf{ctx}} \cdot pk, \mathsf{N}) = e(g_1, g_2)$. Our key insight is that this pairing equation fundamentally verifies a multiplicative relationship. When we expand the left side: $e(g_1^{\mathsf{ctx}} \cdot g_1^k, g_2^{1/(k+\mathsf{ctx})}) = e(g_1, g_2)$ we are effectively proving that $(k + \mathsf{ctx}) \cdot \frac{1}{k+\mathsf{ctx}} = 1$.

This observation suggests an alternative approach: instead of using pairings to verify this relationship, we can prove it directly through a carefully constructed $\Sigma$-protocol. Let:

- $m_1 = k + \mathsf{ctx}$ (committed in $\mathsf{cm}_1$)
- $m_2 = \frac{1}{k+\mathsf{ctx}}$ (committed in $\mathsf{cm}_2$)

The VRF nullifier is then simply $\mathsf{N} = g^{m_2}$, and verification reduces to proving:

1. $m_1$ is correctly formed from committed values $k$ and $\mathsf{ctx}$
2. $m_1 \cdot m_2 = 1$ (multiplicative inverse relation)

3. $\mathsf{N} = g^{m_2}$ (nullifier structure)

This reformulation eliminates the need for pairings while maintaining the security properties of the original VRF. The challenge now becomes constructing an efficient $\Sigma$-protocol that proves these relationships without revealing the underlying values.

## 9.5 Commitment Structure for VRF Verification

To privately prove the VRF relationship, we commit to both the input relationship and multiplicative inverse:

– Primary commitments to inputs:

$$\mathsf{cm}_k = g^k h^{r_1}, \quad \mathsf{cm}_{\mathsf{ctx}} = g^{\mathsf{ctx}} h^{r_2}$$

– Derived commitment to their sum:

$$\mathsf{cm}_3 = \mathsf{cm}_k^{\mathsf{ctx}} h^{r_3}$$

– Commitment to the inverse:

$$\mathsf{cm}_4 = \mathsf{cm}^{m_2} h^{r_4} \text{ where } m_2 = \frac{1}{m_1}$$

The algebraic structure of these commitments enables our $\Sigma$-protocol to efficiently prove the multiplicative inverse relationship while maintaining zero-knowledge.

## 9.6 $\Sigma$-Protocol Construction

Given these commitments, we construct a $\Sigma$-protocol that proves the VRF relationship in zero-knowledge. The protocol leverages auxiliary commitments $\mathsf{cm}_3, \mathsf{cm}_4$ to enforce the multiplicative inverse relationship: $\Pi^{\mathcal{R}_{\mathsf{VRF}}}$ from

$$\mathcal{R}_{\mathsf{vrf}} = \left\{ (\mathsf{cm}_k, \mathsf{cm}_{\mathsf{ctx}}, \mathsf{N}), (k, \mathsf{ctx}, r_1, r_2) \;\middle|\; \mathsf{cm}_k = g^k h^{r_1} \;\wedge\; \mathsf{cm}_{\mathsf{ctx}} = g^{\mathsf{ctx}} h^{r_2} \wedge \mathsf{N} = g^{1/(k+\mathsf{ctx})} \right\}$$

**Protocol 1: VRF Output Verification**

**Common Input:** Commitments $\mathsf{cm}_1, \mathsf{cm}_2$, group generator $g$, and public parameters $h \in \mathbb{G}$

**Prover Input:** Witness $(m_1, m_2, r_1, r_2, r_3, r_4)$ such that:

- $\mathsf{cm}_1 = g^{m_1} h^{r_1}$ and $\mathsf{cm}_2 = g^{m_2} h^{r_2}$
- $\mathsf{cm}_3 = \mathsf{cm}_1^{m_2} h^{r_3}$ and $\mathsf{cm}_4 = h^{r_4}$
- $m_1 \cdot m_2 = 1$ (multiplicative inverse relation)

1. **Commitment:** Prover samples randomness:

$$\alpha_1, \alpha_2, \rho_1, \rho_2, \rho_3, \rho_4 \leftarrow\!\!\$\ \mathbb{Z}_q$$

   Computes:
   - $T_1 \leftarrow g^{\alpha_1} h^{\rho_1}$
   - $T_2 \leftarrow g^{\alpha_2} h^{\rho_2}$
   - $T_3 \leftarrow \mathsf{cm}_1^{\alpha_2} h^{\rho_3}$
   - $T_4 \leftarrow h^{\rho_4}$

   Sends $(T_1, T_2, T_3, T_4)$ to verifier.
2. **Challenge:** Verifier samples $c \leftarrow\!\!\$\ \mathbb{Z}_q$ and sends to prover.
3. **Response:** Prover computes:

$$\begin{aligned} s_1 &\leftarrow \alpha_1 + c \cdot m_1 & u_1 &\leftarrow \rho_1 + c \cdot r_1 \\ s_2 &\leftarrow \alpha_2 + c \cdot m_2 & u_2 &\leftarrow \rho_2 + c \cdot r_2 \\ u_3 &\leftarrow \rho_3 + c \cdot r_3 & u_4 &\leftarrow \rho_4 + c \cdot r_4 \end{aligned}$$

   Sends $(s_1, s_2, u_1, u_2, u_3, u_4)$ to verifier.
4. **Verification:** Verifier checks:
   (i) $g^{s_1} h^{u_1} \stackrel{?}{=} T_1 \cdot \mathsf{cm}_1^c$
   (ii) $g^{s_2} h^{u_2} \stackrel{?}{=} T_2 \cdot \mathsf{cm}_2^c$
   (iii) $\mathsf{cm}_1^{s_2} h^{u_3} \stackrel{?}{=} T_3 \cdot \mathsf{cm}_3^c$
   (iv) $h^{u_4} \stackrel{?}{=} T_4 \cdot \mathsf{cm}_4^c$
   (v) $\frac{\mathsf{cm}_3}{\mathsf{cm}_4} \stackrel{?}{=} g$

*Security Analysis:* The protocol satisfies the following security properties:

- **Completeness:** For honest prover and verifier, all verification equations hold algebraically:

$$\begin{aligned} g^{s_1}h^{u_1} &= g^{\alpha_1+cm_1}h^{\rho_1+cr_1} = T_1 \cdot \mathsf{cm}_1^c \\ g^{s_2}h^{u_2} &= g^{\alpha_2+cm_2}h^{\rho_2+cr_2} = T_2 \cdot \mathsf{cm}_2^c \\ \mathsf{cm}_1^{s_2}h^{u_3} &= \mathsf{cm}_1^{\alpha_2+cm_2}h^{\rho_3+cr_3} = T_3 \cdot \mathsf{cm}_3^c \\ h^{u_4} &= h^{\rho_4+cr_4} = T_4 \cdot \mathsf{cm}_4^c \end{aligned}$$

- **Special Soundness:** Given two accepting transcripts $(T_1, T_2, T_3, T_4, c, s_1, s_2, u_1, u_2, u_3, u_4)$ and $(T_1, T_2, T_3, T_4, c', s_1', s_2', u_1', u_2', u_3', u_4')$ with $c \neq c'$, the extractor $\mathcal{E}$ works as follows:

$$\begin{aligned} m_1 &= \frac{s_1 - s_1'}{c - c'} & r_1 &= \frac{u_1 - u_1'}{c - c'} \\ m_2 &= \frac{s_2 - s_2'}{c - c'} & r_2 &= \frac{u_2 - u_2'}{c - c'} \\ r_3 &= \frac{u_3 - u_3'}{c - c'} & r_4 &= \frac{u_4 - u_4'}{c - c'} \end{aligned}$$

  The extracted witness satisfies all verification equations and the multiplicative inverse relation by the binding property of Pedersen commitments.

- **Honest-Verifier Zero-Knowledge:** The simulator $\mathcal{S}$ operates as follows:

  1. Sample $s_1, s_2, u_1, u_2, u_3, u_4 \leftarrow\$ \mathbb{Z}_q$ uniformly
  2. Compute simulated commitments:

$$\begin{aligned} T_1 &\leftarrow g^{s_1}h^{u_1} \cdot \mathsf{cm}_1^{-c} \\ T_2 &\leftarrow g^{s_2}h^{u_2} \cdot \mathsf{cm}_2^{-c} \\ T_3 &\leftarrow \mathsf{cm}_1^{s_2}h^{u_3} \cdot \mathsf{cm}_3^{-c} \\ T_4 &\leftarrow h^{u_4} \cdot \mathsf{cm}_4^{-c} \end{aligned}$$

  3. Output $(T_1, T_2, T_3, T_4, c, s_1, s_2, u_1, u_2, u_3, u_4)$

  The simulated transcript is perfectly indistinguishable from a real transcript as the distribution of responses $(s_1, s_2, u_1, u_2, u_3, u_4)$ is uniform in both cases, and the commitments are uniquely determined by the verification equations.

*Connection to VRF Security* The protocol's soundness guarantees that $\mathsf{N} = g^{m_2}$ is valid only if $m_2 = 1/m_1$ for $m_1 = k + \mathsf{ctx}$. This is critical because:

- **Pseudorandomness:** Under $q$-DHI, $g^{1/m_1}$ is indistinguishable from random without knowledge of $m_1$.
- **Uniqueness:** The equation $m_1 \cdot m_2 = 1$ has a unique solution in $\mathbb{Z}_p^*$, preventing adversarial equivocation.

Thus, the security of the VRF directly reduces to the hardness of computing discrete logarithms and the soundness of the inverse proof.

**Theorem 6 (VRF Correctness via Multiplicative Inverses).** *Under the discrete logarithm assumption, proving the equality $m_1 \cdot m_2 = 1$ for commitments $\mathsf{cm}_1 = g^{m_1}h^{r_1}$ and $\mathsf{cm}_2 = g^{m_2}h^{r_2}$ is equivalent to verifying the correctness of the VRF output $\mathsf{Nullifier} = g^{m_2}$. Specifically:*

- *Completeness: A valid VRF output always satisfies $m_1 \cdot m_2 = 1$.*
- *Soundness: Any adversary forging a nullifier must break either the binding of the commitments or the q-DHI assumption.*

*Proof (Sybil Resistance Sketch).*

- **Uniqueness:** Direct consequence of the VRF's unique provability. If two nullifiers $\mathsf{N}_0, \mathsf{N}_1$ exist for the same $(k, \mathsf{ctx})$, then $m_1 \cdot m_2^{(0)} = m_1 \cdot m_2^{(1)} = 1$, violating the uniqueness of inverses in $\mathbb{Z}_p^*$.
- **Unlinkability:** Follows from the zero-knowledge property of the $\Sigma$-protocol. The proof reveals only the validity of $m_1 \cdot m_2 = 1$, not $m_1$ or $m_2$.
- **Soundness:** By the extractability of the $\Sigma$-protocol, any valid nullifier must satisfy $\mathsf{N} = g^{1/(k+\mathsf{ctx})}$. Binding of Pedersen commitments ensures $m_1 = k + \mathsf{ctx}$, thus $\mathsf{N}$ is uniquely tied to the credentials.

### 9.7 Performance Evaluation

Dodis Yampolskiy

- Gen = 1 exponentiation
- GT exponentiation, g2 exponentiation
- 1 g1 mul + 2 pairing + 2 GT equality check

## 10 Identity System

### 10.1 Construction

### 10.2 Master Credential Protocol

$\underline{\mathsf{UserKeyGen}(1^\lambda)}$ : $\mathsf{usk}, k_1 \leftarrow\!\!\$\ \mathbb{Z}_p^2$, Return $\mathsf{usk}, k$

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

$\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1) = \mathsf{ZKPoK}\{(\mathsf{cm}_1, \mathsf{usk}, k_1) | \mathsf{cm}_1 = g_1^0 g_2^{k_1} g_2^0 g^{\mathsf{usk}}\}$

$\underline{\mathsf{Obtain}(\mathsf{usk}_\mathsf{m}, \mathsf{opk}_\mathsf{m})}$ $\qquad\qquad\qquad\qquad$ $\underline{\mathsf{Issue}(\mathsf{cm}_\mathsf{m}, \tilde{\mathsf{m}}, \mathsf{osk}_\mathsf{m})}$

$\mathsf{cm}_1 = \mathsf{CM.Com}([0, k, 0]; \mathsf{usk})$ $\quad \xrightarrow{\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)} \quad$ If $\Pi^{\mathcal{R}_{\mathsf{zero}}}(\mathsf{cm}_1)$ fails, return $\perp$

$k_2 \leftarrow\!\!\$\ \mathbb{Z}_p, \mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, k_2, \mathsf{ctx}]; 0),\ \mathsf{cm}_\mathsf{m} = \mathsf{cm}_1 \cdot \mathsf{cm}_2$

If $\mathsf{cm}_\mathsf{m} \neq \mathsf{CM.Com}([\mathsf{id}, k_1 + k_2\mathsf{ctx}]; \mathsf{usk})$ or $\quad \xleftarrow{\mathsf{cm}_\mathsf{m}, \mathsf{id}, k_2, \mathsf{ctx}, \sigma_\mathsf{m}} \quad$ $u \leftarrow\!\!\$\ \mathbb{Z}_p,\ \sigma_\mathsf{m} \leftarrow\!\!\$\ \mathsf{RS.Sign}(\mathsf{cm}_\mathsf{m}, \mathsf{osk}, u)$

If $\mathsf{RS.Ver}(\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{opk}) = 0$, return $\perp$

Else, return $\mathsf{cred}_\mathsf{m} \leftarrow (\sigma_\mathsf{m}, \mathsf{cm}_\mathsf{m}, \mathsf{opk}_\mathsf{m})$

Fig. 5: Private Identity System

## 10.3 Context Credential Protocol

The context credential issuance protocol enables a user to obtain a context credential linked to their root credential. The Context Credential $\mathsf{cred_c}$ contains a signature $\sigma_\mathsf{c}$ over commitment $\mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ issued by $\mathcal{I}_\mathsf{c}$

$\underline{(\mathsf{OrgKeyGen}, \mathsf{UserKeyGen}):}$ proceed analogously to Master Credential

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

$\Pi^{\mathcal{R}_\mathsf{verkey}}$ proceed analogously to Master Credential

$\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}, \phi) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}) | \mathsf{cm_c} = g_1^{\mathsf{id}} g_2^{\mathsf{ctx}} g^{\mathsf{usk_c}} \wedge \mathsf{ctx} = \text{"}context\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{m}', \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}master\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Obtain}(\mathsf{usk_c}, \mathsf{opk_c}, \vec{m})}$ $\underline{\mathsf{Issue}(\mathsf{cm_c}, \mathsf{osk_c})}$

$$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c}), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{s.disclose}}(\mathsf{cm_c})$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c})$ fails, return $\perp$.

If $\mathsf{cm_c} \neq \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c})$ or $\xleftarrow{\mathsf{cm_c}, \mathsf{id}, \mathsf{ctx}, \sigma_\mathsf{c}}$ Else, $u \leftarrow\!\!\$\ \mathbb{Z}_p$, $\sigma_\mathsf{c} \leftarrow\!\!\$\ \mathsf{RS.Sign}(\mathsf{cm_c}, \mathsf{osk_c}, u)$

If $\mathsf{RS.Ver}(\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c}) = 0$, return $\perp$

Else, return $\mathsf{cred_c} \leftarrow (\sigma_\mathsf{c}, \mathsf{cm_c}, \mathsf{opk_c})$

$\underline{(\mathsf{Show}, \mathsf{Verify}):}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_m}') | \mathsf{cm_m}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_m}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{m}', \mathsf{opk_m}) = 1 \wedge \mathsf{ctx} = \text{"}master\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}') = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx}, \mathsf{usk_c}') | \mathsf{cm_c}' = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx}]; \mathsf{usk_c}') \wedge \mathsf{RS.Ver}(\sigma_\mathsf{c}', \mathsf{opk_c}) = 1 \wedge \mathsf{ctx} = \text{"}context\text{"}\}$

$\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}) = \mathsf{ZKPoK}\{(\mathsf{id}, \mathsf{ctx_m}, \mathsf{ctx_c}, \mathsf{usk_m}, \mathsf{usk_c}) | \mathsf{cm_m} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_m}]; \mathsf{usk_m}) \wedge \mathsf{cm_c} = \mathsf{CM.Com}([\mathsf{id}, \mathsf{ctx_c}]; \mathsf{usk_c})\}$

$\underline{\mathsf{Show}(\mathsf{cred_m})}$ $\underline{\mathsf{Verify}(\sigma_\mathsf{m}', \mathsf{cm_m}', \mathsf{opk_m})}$

Compute $\mathsf{cred_m}', \mathsf{cred_c}'$ analogously to Master Credential

$$\xrightarrow{\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}'), \Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}'), \Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')}$$

If $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_m}')$, or $\Pi^{\mathcal{R}_\mathsf{sok}}(\mathsf{cred_c}')$, or $\Pi^{\mathcal{R}_\mathsf{eq}}(\mathsf{cm_m}', \mathsf{cm_c}')$ fails, return 0, Else 1

Fig. 6: Context Credential Protocol

## 11 Summary of Zero-Knowledge Proof Concepts

In the context of zero-knowledge proofs (ZKPs) for our multi-issuer, multi-credential anonymous credential system, we define the following key concepts to clarify their roles in our security definitions:

- **Relation $\mathcal{R}$**: A binary relation $\mathcal{R}$ defines a set of valid pairs $(x, w)$, where:

  - $*$ $x$ is the *statement* (public claim or condition to be proven).

  - $*$ $w$ is the *witness* (private information, e.g., attributes or secret keys).

  $\mathcal{R}$ specifies the condition under which a statement $x$ is true, i.e., $(x, w) \in \mathcal{R}$ if $w$ is a valid witness for $x$. It is not the predicate but the underlying condition for validity.

- **Statement** $x$: The public input to the proof system, representing the claim to be proven. In our system, $x$ includes:

  - $*$ A credential $\mathsf{cred}$

  - $*$ A commitment to attributes $\mathsf{cm}$

  - $*$ A predicate $\phi$, specifying a condition (e.g., "age $>$ 18") the attributes must satisfy.

  The statement $x$ is part of the relation's input, not the relation itself.

- **Predicate** $\phi$: A specific condition or policy within the statement $x$ that the witness (e.g., attributes $m$ must satisfy. It is not generated by the prover but is part of the public statement provided to the verifier.

- **Witness** $w$: Private information, such as attributes $m$ and commitment opening (e.g., $\mathsf{usk}[i]$), that satisfies $\mathcal{R}$ for a given $x$.

- **ZKP Process**:

  - $*$ The prover generates a proof $\pi$ using:

    $$\text{Prove}(x, w) \to \pi$$

    where $x$ includes $\phi$, and sends $\pi$ and $x$ to the verifier.

  - $*$ The verifier checks the proof using:

    $$\text{Verify}(\pi, x) \to 0/1$$

    outputting 1 if the proof is valid, 0 otherwise.

In our unforgeability game, the adversary outputs $\mathsf{cred}^*, \mathsf{cm}, \pi^*, \phi^*$ where $\phi^*$ is the predicate in the forged statement $x = \mathsf{cred}^*, \mathsf{cm}^*, \phi^*)$, and $\pi^*$ proves the existence of a witness $w$ satisfying the relation $\mathcal{R}$ for $x$.

## 12 Zero-Knowledge Proofs

Zero-Knowledge Proofs (ZKPs) enable a prover $\mathcal{P}$ to demonstrate the validity of a statement $x$ while keeping the corresponding witness $w$ secret. In the context of anonymous credentials, ZKPs allow a user to prove possession of a credential, verify the validity of its attributes, and comply with related policies, all while revealing none or only minimal sensitive underlying data.

Efficient Zero-Knowledge Proofs are a fundamental building block of anonymous credential systems. We show that by leveraging the modularity and composability of ZKPs, we construct a suite of protocols that form the backbone of our credential system and intertwine with the security properties of our other components to achieve the fundamental properties of Anonymous Credentials - Correctness, Unforgeability, and Anonymity.

We organize our discussion in three parts

1. We first introduce the fundamental concepts of zero-knowledge proofs and the core protocols, including interactive proof systems, proofs of knowledge, and $\Sigma$-protocols.
2. Next, we extend the discussion to more complex protocols such as compound statements and pairing-based constructs that are commonly used in anonymous credential systems.
3. Finally, we present our original contribution: a novel proof of multiplicative inverse. We demonstrate how this protocol is used to prove the correct evaluation of a Verifiable Random Function (VRF) created from secret messages in two distinct commitments, and we provide a detailed analysis comparing its efficiency and performance to previous and alternate work.

In the following section, we will leverage these constructions with our rerandomizable signature over commitments to build a multi-issuer, multi-credential anonymous credential system with sybil resistance.

### 12.1 Interactive Proof Systems, the Prover $\mathcal{P}$ and Verifier $\mathcal{V}$

An interactive proof system for a language $L$ consists of two probabilistic interactive Turing machines $(\mathcal{P}, \mathcal{V})$, where $\mathcal{P}$ is the prover and $\mathcal{V}$ is the verifier. In the context of Anonymous Credentials, $\mathcal{P}$ represents a user holding a credential issued by an Issuer, whereas $\mathcal{V}$ represents any public system tasked with verifying the user's identity and attributes.

The interactive proof system must satisfy the following properties:

- **Completeness**: For all $x \in L$, if both parties follow the protocol honestly, then the verifier accepts with overwhelming probability:
$$\Pr[(\mathcal{P}, \mathcal{V})(x) = 1] \geq 1 - \epsilon(|x|),$$
where $\epsilon(|x|)$ is a negligible function. This property guarantees that honest interactions will almost always lead to acceptance.
- **Soundness**: For all $x \notin L$ and any (potentially malicious) prover $\mathcal{P}^*$, the probability that $\mathcal{V}$ accepts is negligible:
$$\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] \leq \delta(|x|),$$
where $\delta(|x|)$ is a negligible function. This ensures that any attempt by a dishonest prover to convince the verifier of a false statement will be unsuccessful except with negligible probability.

### 12.2 Proofs of Knowledge, the Extractor $\mathcal{E}$, and Signatures of Knowledge

In many cryptographic settings, including anonymous credential systems, it is not sufficient to simply prove that a statement is valid; one must also ensure that the prover genuinely possesses the underlying secret. This requirement is critical for enforcing non-transferability of credentials and for enabling constructs like signatures of knowledge, where the user not only signs a statement but demonstrates knowledge of a hidden secret key or witness.

A proof of knowledge introduces an extractor $\mathcal{E}$, a theoretical algorithm that—given the ability to rewind a prover $\mathcal{P}^*$ that convinces the verifier with non-negligible probability—can efficiently extract the corresponding witness $w$. Formally, for any prover $\mathcal{P}^*$,

$$\Pr\left[\mathcal{E}^{\mathcal{P}^*}(x) = w : (x, w) \in R\right] \geq \Pr\left[(\mathcal{P}^*, \mathcal{V})(x) = 1\right] - \nu(|x|),$$

where $\nu(|x|)$ is negligible and $R$ is the underlying relation.

This property is essential when constructing a signature of knowledge, where the signer must prove, in zero-knowledge, that they possess the secret value corresponding to their public credential. The extractor's role here guarantees that any valid proof (or signature) could only have been produced by someone who actually knows the secret witness. In other words, an adversary cannot forge a valid signature-of-knowledge without having genuine knowledge of the secret, which is crucial for preventing credential sharing or transfer.

### 12.3 Zero-Knowledge and the Simulation $\mathcal{S}$ Paradigm

Zero-knowledge strengthens interactive proofs through the simulation paradigm. The key insight is that if a simulator $\mathcal{S}$ can produce transcripts indistinguishable from real protocol interactions without access to the witness, then the protocol reveals no knowledge beyond the statement's validity.

Formally, for all probabilistic polynomial-time verifiers $\mathcal{V}^*$, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for all $x \in L$:

$$\{\text{VIEW}_{\mathcal{V}^*}(\mathcal{P}, \mathcal{V}^*)(x)\} \approx_c \{\mathcal{S}(x)\}$$

where VIEW represents the verifier's view of the real protocol interaction and $\approx_c$ denotes computational indistinguishability.

Informally, the simulator paradigm provides a powerful proof technique for demonstrating zero-knowledge. If an efficient algorithm (the simulator) can generate protocol transcripts without access to any witness (the secret), and these transcripts are computationally indistinguishable from real protocol executions, that is, they look the same, then the verifier must not be able to learn anything meaningful about the witness during actual protocol runs. This is because anything the verifier could compute from a real interaction could equally well be computed from the simulated transcripts. The simulator effectively demonstrates that the verifier's view of the protocol can be reconstructed without any knowledge of the secret witness, a concept first introduced by Goldwasser, Micali, and Rackoff in their seminal 1985 paper.

### 12.4 Zero-Knowledge and the Simulation Paradigm

Zero-knowledge proofs ensure that no additional information is leaked during the proof process beyond the validity of the statement being proven. The central idea behind this guarantee is captured by the simulation paradigm. In this paradigm, one demonstrates that for any efficient (i.e., probabilistic polynomial-time) verifier, there exists a simulator $\mathcal{S}$ that can produce transcripts of interactions that are computationally indistinguishable from those produced in an actual honest execution of the protocol.

Formally, for every probabilistic polynomial-time verifier $\mathcal{V}^*$ and for every statement $x \in L$, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that the verifier's view generated during its interaction with the honest prover $\mathcal{P}$,

$$\{\text{VIEW}_{\mathcal{V}^*}(\mathcal{P}, \mathcal{V}^*)(x)\},$$

is computationally indistinguishable from the output of the simulator,

$$\{\mathcal{S}(x)\}.$$

We denote this equivalence by:

$$\{\text{VIEW}_{\mathcal{V}^*}(\mathcal{P},\mathcal{V}^*)(x)\} \approx_c \{\mathcal{S}(x)\}.$$

The fundamental implication of this property is that, since the simulator can produce convincing transcripts without access to any witness, the verifier gains no additional knowledge about the secret; the simulator demonstrates that every piece of information the verifier observes could have been generated independently, hence ensuring the zero-knowledge property.

## 12.5 $\Sigma$-Protocol Template

A $\Sigma$-protocol follows a specific three-move structure, Commit, Challenge, Response:

**Protocol 2: $\Sigma$-Protocol Template**

**Common Input:** Statement $x$

**Prover Input:** Witness $w$ such that $(x, w) \in R$

1. **Commitment:** $\mathcal{P}$ computes first message $a$ (the commitment) and sends it to $\mathcal{V}$
2. **Challenge:** $\mathcal{V}$ sends a random $t$-bit challenge $e$
3. **Response:** $\mathcal{P}$ computes response $z$ and sends it to $\mathcal{V}$
4. $\mathcal{V}$ accepts if verification predicate $\phi(x, a, e, z) = 1$

## 12.6 $\Sigma$-Protocol Framework

A $\Sigma$-protocol for relation $R$ is a three-move public-coin protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$ that follows our template structure while satisfying three properties

**Definition 28 ($\Sigma$-Protocol).** *A protocol $\pi$ is a $\Sigma$-protocol for relation $R$ if it is a three-move public-coin protocol and satisfies:*

- ***Completeness:*** *If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on common input $x$ and private input $w$ to $\mathcal{P}$ where $(x, w) \in R$, then $\mathcal{V}$ always accepts.*
- ***Special Soundness:*** *There exists a polynomial-time extractor $\mathcal{E}$ that, given any $x$ and any pair of accepting transcripts $(a, e, z)$ and $(a, e', z')$ where $e \neq e'$, outputs $w$ such that $(x, w) \in R$.*
- ***Special Honest-Verifier Zero-Knowledge:*** *There exists a polynomial-time simulator $\mathcal{S}$ that, on input $x$ and challenge $e$, outputs a transcript $(a, e, z)$ with the same probability distribution as transcripts between honest $\mathcal{P}(x, w)$ and $\mathcal{V}(x, e)$.*

Special soundness requires extraction from any pair of accepting transcripts with the same first message, which is stronger than standard knowledge extraction. Similarly, special HVZK requires simulation for any given challenge, not just a random one.

## 12.7 Schnorr's Zero Knowledge Proof of Knowledge

The canonical example of a $\Sigma$-protocol is Schnorr's identification scheme which demonstrates proof of knowledge of a discrete logarithm, where a prover convinces a verifier they know $w$ such that $h = g^w$ without revealing $w$.

For a cyclic group $\mathbb{G}$ of prime order $q$ with generator $g$, the relation is formally defined as:

$$\mathcal{R}_{\mathsf{DL}} = \{(h, w) \in \mathbb{G} \times \mathbb{Z}_q : h = g^w\}$$

$\mathcal{R}_{\mathsf{DL}}$ captures the statement $x$, that $h$ is the public key derived from secret key $w$.

**Protocol 3: Schnorr's Protocol**

**Common Input:** Group element $h \in \mathbb{G}$

**Prover Input:** $w \in \mathbb{Z}_q$ such that $h = g^w$

1. **Commitment:** $\mathcal{P}$ samples $r \leftarrow\$ \mathbb{Z}_q$, computes $a = g^r$
2. **Challenge:** $\mathcal{V}$ samples challenge $e \leftarrow\$ \{0,1\}^t$ where $2^t < q$
3. **Response:** $\mathcal{P}$ computes $z = r + ew \mod q$
4. Verification: $\mathcal{V}$ accepts if $g^z = a \cdot h^e$

We show this satisfies $\Sigma$-protocol the 3 properties

- **Completeness:** If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on common input $x$ and private input $w$ to $\mathcal{P}$ where $(x, w) \in R$, then $\mathcal{V}$ always accepts.
- **Special Soundness:** Given transcripts $(a, e, z)$ and $(a, e', z')$ where $e \neq e'$, one can efficiently extract $w = \frac{(z-z')}{(e-e')} \mod q$
- **Special Honest-Verifier Zero-Knowledge:** For any fixed challenge $e$, there exists a simulator $\mathcal{S}$ that can efficiently simulate accepting transcripts by sampling $z \leftarrow\$ \mathbb{Z}_q$ and computing $a = g^z h^{-e}$

**Notation**

## 12.8 Relations and Languages

We begin by defining the fundamental notion of a relation and its associated language. For a relation $\mathcal{R}$, we define its language $L_{\mathcal{R}}$ as:

$$L_{\mathcal{R}} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$$

where $x$ represents the statement to be proven and $w$ represents the witness.

## 12.9 Basic Cryptographic Relations

The discrete logarithm relation, which forms the basis for many of our constructions, is defined as:

$$\mathcal{R}_{\mathsf{DL}} = \{((g, h), w) \in (\mathbb{G} \times \mathbb{G}) \times \mathbb{Z}_q : h = g^w\}$$

where $g$ is explicitly included as part of the statement. In practice, when $g$ is a fixed system parameter, we often use the simplified notation:

$$\mathcal{R}_{\mathsf{DL}} = \{(h, w) : h = g^w\}$$

For Pedersen commitments, where both the commitment and its opening are part of the proof, we define:

$$\mathcal{R}_{\mathsf{com}} = \{(\mathsf{cm}, (m, r)) : \mathsf{cm} = g^m h^r\}$$

When proving relations between multiple commitments, we extend this notation. For example, proving equality of committed values:

$$\mathcal{R}_{\mathsf{eq}} = \{(\mathsf{cm}_1, \mathsf{cm}_2), (m, r_1, r_2)) : \mathsf{cm}_1 = g^m h^{r_1} \wedge \mathsf{cm}_2 = g^m h^{r_2}\}$$

### 12.10 Relations and Languages

A central concept in constructing zero-knowledge protocols is that of a relation and its corresponding language. In cryptographic proofs, a relation $\mathcal{R}$ is defined as a subset of pairs $(x, w)$ where $x$ is a statement and $w$ is the associated witness such that a predetermined predicate $\phi$ is satisfied. The language associated with $\mathcal{R}$, denoted $L_{\mathcal{R}}$, is defined as:

$$L_{\mathcal{R}} = \{x \mid \exists w \text{ such that } (x, w) \in \mathcal{R}\}.$$

This abstraction enables us to capture the notion of "proof of a statement" as a proof of membership in a language.

For example, consider the discrete logarithm relation, which is a key building block in many protocols:

$$\mathcal{R}_{\mathsf{DL}} = \{(h, w) \in \mathbb{G} \times \mathbb{Z}_q : h = g^w\}.$$

Here, $x = h$ is the public statement and $w$ is the discrete logarithm relative to the generator $g$.

Similarly, for commitment schemes, notably the Pedersen commitment, the relation is captured as:

$$\mathcal{R}_{\mathsf{com}} = \{(\mathsf{cm}, (m, r)) : \mathsf{cm} = g^m h^r\}.$$

In this context, the language consists of all valid commitments, and the corresponding witness includes both the message $m$ and the randomness $r$.

## 13 Advanced Proof Protocols

Advanced proof protocols are extensions of basic proof protocols

Proof protocols are combined with the properties of other building blocks to enable

Correctness Unforgeability Anonymity

1. Proof of Knowledge (PoK) for the opening of commitments.
2. PoK demonstrating linear relations (e.g., that the sum of certain exponents equals a separate exponent).
3. PoK of Zero values.
4. Signatures of Knowledge that seamlessly integrate both signing and proof of knowledge.
5. Multiple commitments and equality proofs for exponents, crucial for proving consistent userid across various credentials.

Following this, we will build upon these foundational constructs to introduce our novel contribution: a proof of multiplicative inverse that further enhances the efficiency and expressiveness of verifiable random functions derived from multiple commitments.

Our work underscores that the confluence of these individual security properties forms the backbone of secure anonymous credential systems, enabling a rich set of functionalities while ensuring stringent privacy guarantees.

### 13.1 Protocol Categories and Their Roles

### 13.2 Structure and Flow

### 13.3 Proof of Knowledge for Pedersen Commitment Opening

In anonymous credential systems, to verify a credential, rather than sending the messages in plainsight, a prover demonstrates knowledge of the opening of a Pedersen commitment without revealing the message or randomness.

$$\mathsf{cm} = g^m h^r \text{ and often denoted in shorthand } \mathsf{CM.Com}([m]; r)$$

where $g, h \in \mathbb{G}$ (a cyclic group of prime order $q$), $m$ is the message, and $r$ is the randomness. The corresponding relation is

$$\mathcal{R}_{\mathsf{com}} = \{(\mathsf{cm}, (m, r)) \mid \mathsf{cm} = g^m h^r\}.$$

**Protocol 4: Pedersen Commitment Opening**

**Common Input:** Commitment $\mathsf{cm}$ and public parameters $g, h \in \mathbb{G}$

**Prover Input:** Witness $(m, r)$ such that $\mathsf{cm} = g^m h^r$

1. **Commitment:** Prover samples $\alpha, \rho \leftarrow\$ \mathbb{Z}_q$, computes
$$T \leftarrow g^\alpha h^\rho,$$
and sends $T$ to the verifier.
2. **Challenge:** Verifier samples a challenge $c \leftarrow\$ \mathbb{Z}_q$ and sends it to the prover.
3. **Response:** Prover computes
$$s \leftarrow \alpha + c\,m \quad \text{and} \quad u \leftarrow \rho + c\,r,$$
and sends $(s, u)$ to the verifier.
4. **Verification:** Verifier checks if
$$g^s h^u \overset{?}{=} T \cdot \mathsf{cm}^c.$$

*Security Properties:* This $\Sigma$-protocol template satisfies:

- **Completeness:** Honest execution leads the verifier to accept.
- **Special Soundness and the Extractor:** Assume that an adversarial prover produces two accepting transcripts for the same commitment $T$ but with distinct challenges $c$ and $c'$:
$$(T, c, s, u) \quad \text{and} \quad (T, c', s', u'),$$
with $c \neq c'$. The extractor $\mathcal{E}$ operates by rewinding the prover to the point immediately after sending $T$ and then issuing two different challenges $c$ and $c'$. Given that in an honest execution the responses satisfy
$$s = \alpha + c\,m \quad \text{and} \quad u = \rho + c\,r,$$
we have from the two transcripts:
$$s - s' = (c - c')\,m \quad \text{and} \quad u - u' = (c - c')\,r.$$
Thus, the extractor computes the witness as:
$$m = \frac{s - s'}{c - c'} \quad \text{and} \quad r = \frac{u - u'}{c - c'}.$$

This demonstrates that any prover capable of producing two valid responses for different challenges must, in fact, know the correct opening $(m, r)$ of the commitment, thereby ensuring the protocol's special soundness.

- **Honest-Verifier Zero-Knowledge:** We proceed with the simulator construction as follows:

  Let $\mathcal{S}$ be a simulator that, given the public input $\mathsf{cm}$ and a challenge $c \in \mathbb{Z}_q$, produces a transcript indistinguishable from one generated in an honest run of the protocol. The simulator works as follows:

  1. **Random Response Generation:** $\mathcal{S}$ uniformly samples $s, u \leftarrow\!\!\$\ \mathbb{Z}_q$. These will serve as the simulated responses.
  2. **Commitment Computation:** $\mathcal{S}$ computes the simulated commitment message by setting
  $$T \leftarrow g^s h^u \cdot \mathsf{cm}^{-c}.$$
  This computation ensures that during verification,
  $$T \cdot \mathsf{cm}^c = g^s h^u,$$
  thus satisfying the verifier's check.
  3. **Transcript Output:** The simulator outputs the transcript $(T, c, s, u)$.

  *Justification:* In a real protocol execution, the prover computes $T = g^\alpha h^\rho$ with $\alpha$ and $\rho$ chosen uniformly at random, and then calculates $s = \alpha + c\,m$ and $u = \rho + c\,r$. Since $\alpha$ and $\rho$ are uniformly random in $\mathbb{Z}_q$, it follows that $s$ and $u$ are uniformly distributed in $\mathbb{Z}_q$ as well. In the simulation, by sampling $s$ and $u$ uniformly at random, and then setting
  $$T \leftarrow g^s h^u \cdot \mathsf{cm}^{-c},$$
  the simulator produces a transcript with the same distribution as in an honest execution. This construction satisfies the special honest-verifier zero-knowledge property, as the transcript $(T, c, s, u)$ is computationally indistinguishable from transcripts generated during real interactions.

## 13.4 AND proofs for protocol composition

In many applications, such as anonymous credential systems, it is required to prove in zero-knowledge that a prover knows witnesses for two separate statements. Assume we have two relations:
$$\mathcal{R}_1 = \{(x_1, w_1) \mid x_1 \text{ is a valid instance with witness } w_1\}$$
and
$$\mathcal{R}_2 = \{(x_2, w_2) \mid x_2 \text{ is a valid instance with witness } w_2\}.$$

For each relation, a $\Sigma$-protocol exists. For instance, the Pedersen commitment opening protocol is a $\Sigma$-protocol for the relation
$$\mathcal{R}_{\mathsf{com}} = \{(\mathsf{cm}, (m, r)) \mid \mathsf{cm} = g^m h^r\}.$$

Our goal is now to construct an AND proof that convinces a verifier that the prover knows witnesses $w_1$ and $w_2$ such that $(x_1, w_1) \in \mathcal{R}_1$ *and* $(x_2, w_2) \in \mathcal{R}_2$.

**Protocol 5: AND proofs with Pedersen Commitment Openings**

**Common Input:** Commitments $\mathsf{cm}_1, \mathsf{cm}_2$ and public parameters $g, h \in \mathbb{G}$

**Prover Input:** Witnesses $(m_1, r_1)$ and $(m_2, r_2)$ such that

$$\mathsf{cm}_1 = g^{m_1} h^{r_1} \quad \text{and} \quad \mathsf{cm}_2 = g^{m_2} h^{r_2}.$$

1. **Commitment:** Prover samples $\alpha_1, \rho_1, \alpha_2, \rho_2 \leftarrow\!\!\$\ \mathbb{Z}_q$, computes

$$T_1 \leftarrow g^{\alpha_1} h^{\rho_1} \text{ for } \mathsf{cm}_1 \quad \wedge \quad T_2 \leftarrow g^{\alpha_2} h^{\rho_2} \text{ for } \mathsf{cm}_2$$

   sends $T_1, T_2)$ to $\mathcal{V}$

2. **Challenge:** $\mathcal{V}$ samples a challenge $c \leftarrow\!\!\$\ \mathbb{Z}_q$ and sends it to the $\mathcal{P}$.
3. **Response:**
   (a) Prover computes

$$s_1 \leftarrow \alpha_1 + c\, m_1, \quad u_1 \leftarrow \rho_1 + c\, r_1, \text{for } \mathsf{cm}_1$$

$$s_2 \leftarrow \alpha_2 + c\, m_2, \quad u_2 \leftarrow \rho_2 + c\, r_2, \text{for } \mathsf{cm}_2$$

   (b) Prover sends the responses $(s_1, u_1)$ and $(s_2, u_2)$ to the verifier.
4. **Verification:** Verifier accepts if both equations hold

$$g^{s_1} h^{u_1} \stackrel{?}{=} T_1 \cdot \mathsf{cm}_1^c \quad \wedge \quad g^{s_2} h^{u_2} \stackrel{?}{=} T_2 \cdot \mathsf{cm}_2^c$$

This AND proof protocol leverages the security properties of the individual $\Sigma$-protocols. By using a common challenge $c$ for both subprotocols, the overall protocol maintains:

- **Completeness,** since honest executions produce valid transcripts;
- **Special Soundness,** as the extraction of both witnesses is possible from two transcripts with differing challenges; and
- **Honest-Verifier Zero-Knowledge,** because a simulator can generate indistinguishable transcripts.

This construction can be extended to other Boolean combinations (such as OR proofs) and alternative cryptographic relations, ensuring robust and modular design in advanced cryptographic protocols.

## 13.5 Proof of Knowledge for Linear Relations

In anonymous credential systems, it is often necessary to prove that committed values satisfy linear relationships (e.g., $m_3 = m_1 + m_2$) without revealing the individual messages or randomness. Let $\mathsf{cm}_1 = g^{m_1} h^{r_1}$, $\mathsf{cm}_2 = g^{m_2} h^{r_2}$, and $\mathsf{cm}_3 = g^{m_1+m_2} h^{r_3}$. The relation is defined as:

$$\mathcal{R}_{\mathsf{linear}} = \left\{ \big((\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_3), (m_1, m_2, r_1, r_2, r_3)\big) \;\middle|\; \mathsf{cm}_1 = g^{m_1} h^{r_1} \wedge \mathsf{cm}_2 = g^{m_2} h^{r_2} \wedge \mathsf{cm}_3 = g^{m_1+m_2} h^{r_3} \right\}.$$

**Protocol 6: Linear Relation Proof**

**Common Input:** $\mathsf{cm}_1, \mathsf{cm}_2, \mathsf{cm}_3$ and public parameters $g, h \in \mathbb{G}$

**Prover Input:** Witness $(m_1, m_2, r_1, r_2, r_3)$ satisfying $\mathcal{R}_{\mathsf{linear}}$

1. **Commitment:** Prover samples $\alpha_1, \alpha_2 \leftarrow\$ \mathbb{Z}_q$ and $\rho_1, \rho_2, \rho_3 \leftarrow\$ \mathbb{Z}_q$, then computes:

$$T_1 \leftarrow g^{\alpha_1} h^{\rho_1}, \quad T_2 \leftarrow g^{\alpha_2} h^{\rho_2}, \quad T_3 \leftarrow g^{\alpha_1+\alpha_2} h^{\rho_3},$$

   and sends $T_1, T_2, T_3$ to the verifier.

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

3. **Response:** Prover computes:

$$s_1 \leftarrow \alpha_1 + c\, m_1, \quad s_2 \leftarrow \alpha_2 + c\, m_2, \quad s_3 \leftarrow \alpha_1 + \alpha_2 + c\,(m_1 + m_2),$$

$$u_1 \leftarrow \rho_1 + c\, r_1, \quad u_2 \leftarrow \rho_2 + c\, r_2, \quad u_3 \leftarrow \rho_3 + c\, r_3,$$

   and sends $(s_1, s_2, s_3, u_1, u_2, u_3)$ to the verifier.

4. **Verification:** Verifier checks:

$$g^{s_1} h^{u_1} \stackrel{?}{=} T_1 \cdot \mathsf{cm}_1^c, \quad g^{s_2} h^{u_2} \stackrel{?}{=} T_2 \cdot \mathsf{cm}_2^c, \quad g^{s_3} h^{u_3} \stackrel{?}{=} T_3 \cdot \mathsf{cm}_3^c.$$

*Security Properties:*

- **Completeness:** Substituting $s_i = \alpha_i + c\, m_i$ and $u_i = \rho_i + c\, r_i$ into the verification equations confirms correctness.

- **Special Soundness:** Let $(T_1, T_2, T_3, c, s_1, s_2, s_3, u_1, u_2, u_3)$ and $(T_1, T_2, T_3, c', s_1', s_2', s_3', u_1', u_2', u_3')$ be two transcripts with $c \neq c'$. The extractor $\mathcal{E}$ computes:

$$m_1 = \frac{s_1 - s_1'}{c - c'}, \quad m_2 = \frac{s_2 - s_2'}{c - c'}, \quad r_i = \frac{u_i - u_i'}{c - c'} \;\; (i \in \{1, 2, 3\}),$$

  and verifies that $\mathsf{cm}_3 = g^{m_1+m_2} h^{r_3}$. This ensures $m_3 = m_1 + m_2$.

- **Honest-Verifier Zero-Knowledge:** A simulator $\mathcal{S}$, given $c$, samples $s_1, s_2, s_3, u_1, u_2, u_3 \leftarrow\$ \mathbb{Z}_q$ and computes:

$$T_1 \leftarrow g^{s_1} h^{u_1} \mathsf{cm}_1^{-c}, \quad T_2 \leftarrow g^{s_2} h^{u_2} \mathsf{cm}_2^{-c}, \quad T_3 \leftarrow g^{s_3} h^{u_3} \mathsf{cm}_3^{-c}.$$

  The output transcript $(T_1, T_2, T_3, c, s_1, s_2, s_3, u_1, u_2, u_3)$ is indistinguishable from a real interaction.

*Justification:* The protocol preserves the linear relationship $m_3 = m_1 + m_2$ by enforcing algebraic consistency across the commitments. The extractor leverages the binding property of Pedersen commitments and the linearity of the exponents to recover the witness. The simulator's ability to retroactively construct valid transcripts ensures no information about $m_1, m_2, r_i$ is leaked.

### 13.6 Proof of Knowledge of Zero Values

In anonymous credential systems, selective disclosure often requires proving that specific committed values are zero without revealing their positions. Let $\mathsf{cm} = \prod_{i=1}^{n} g_i^{m_i} h^r$ be a Pedersen commitment where subset $J \subseteq [n]$ satisfies $m_j = 0$ for $j \in J$. The relation is:

$$\mathcal{R}_{\mathsf{zero}} = \left\{ (\mathsf{cm}, J), (\{m_i\}_{i \notin J}, r) \;\middle|\; \mathsf{cm} = \prod_{i=1}^{n} g_i^{m_i} h^r \wedge m_j = 0 \ \forall j \in J \right\}.$$

**Protocol 7: Zero-Value Proof**

**Common Input:** $\mathsf{cm}$, index set $J$, and public parameters $g_1, \ldots, g_n, h \in \mathbb{G}$

**Prover Input:** Witness $(\{m_i\}_{i \notin J}, r)$ with $m_j = 0$ for $j \in J$

1. **Key Modification:** Prover constructs reduced commitment key $\mathsf{ck}' = (g_i)_{i \notin J} \cup \{h\}$.
2. **Equivalence Proof:** Prover shows $\mathsf{cm}$ can be expressed under $\mathsf{ck}'$:

$$\mathsf{cm} = \prod_{i \notin J} g_i^{m_i} h^r.$$

3. **Standard ZKPoK:** Prover executes a Schnorr-type protocol for $\mathsf{ck}'$ to demonstrate knowledge of $\{m_i\}_{i \notin J}$ and $r$.

**Zero-Value Proof Application to Selective Disclosure** Consider a Pedersen commitment scheme with generators $g_1, g_2, h \in \mathbb{G}$ and a commitment to two messages $m_1, m_2$ with randomness $r$:

$$\mathsf{cm} = g_1^{m_1} g_2^{m_2} h^r.$$

Suppose the prover wishes to disclose $m_1$ while keeping $m_2$ and $r$ private. The protocol proceeds as follows:

**Protocol 8: Selective Disclosure**

**Common Input:** $\mathsf{cm}$, generators $g_1, g_2, h$

**Prover Input:** $(m_1, m_2, r)$ such that $\mathsf{cm} = g_1^{m_1} g_2^{m_2} h^r$

**Disclosed Value:** $m_1$

1. **Disclosure:** Prover sends $m_1$ to the verifier.
2. **Commitment Adjustment:** Verifier computes
$$\mathsf{cm}' \leftarrow \mathsf{cm} \cdot g_1^{-m_1} = g_2^{m_2} h^r.$$
3. **Proof of Knowledge:** Prover executes a ZKPoK for the relation:
$$\mathcal{R}_{\mathsf{hidden}} = \{(\mathsf{cm}', (m_2, r)) \mid \mathsf{cm}' = g_2^{m_2} h^r\}.$$
This is instantiated via a Schnorr-type protocol:
   (a) Prover samples $\alpha, \rho \leftarrow\$ \mathbb{Z}_p$, computes $T \leftarrow g_2^{\alpha} h^{\rho}$, and sends $T$.
   (b) Verifier sends challenge $c \leftarrow\$ \mathbb{Z}_p$.
   (c) Prover computes responses $s \leftarrow \alpha + cm_2$, $u \leftarrow \rho + cr$, and sends $(s, u)$.
   (d) Verifier checks $g_2^s h^u \overset{?}{=} T \cdot (\mathsf{cm}')^c$.

This can be generalized to multiple hidden and public messages.

## 13.7 Two-Party Secret Sharing Protocol

Building on our Pedersen commitment framework (§??), we present a minimal two-party protocol for establishing shared secrets between a user (prover) and issuer. This construction provides the basis for sybil-resistant credential issuance while maintaining privacy.

**Protocol 9: Two-Party Secret Sharing with Zero Knowledge**

**Parties:** User $\mathcal{U}$, Issuer $\mathcal{I}$

**Public Parameters:** Generators $g_1, g_2, h \in \mathbb{G}$, public identifier $\mathsf{id}$

**Goal:** Establish $s = s_1 + s_2$ where $\mathcal{U}$ contributes $s_1$, $\mathcal{I}$ contributes $s_2$

1. **Commitment Phase:**
   - $\mathcal{U}$ chooses $s_1 \leftarrow\!\!\$\ \mathbb{Z}_p$, randomness $r \leftarrow\!\!\$\ \mathbb{Z}_p$, computes:

     $$\mathsf{cm}_1 = \mathsf{CM.Com}([0, s_1]; a) = g_1^0 g_2^{s_1} h^r$$

     and proves in ZKPoK:

     $$\mathcal{R} = \{(\mathsf{cm}_1, (s_1, r)) \mid \mathsf{cm}_1 = g_1^{m_1} g_2^{s_1} h^r \wedge m_1 = 0 \wedge s_1 \neq 0\}$$

   - $\mathcal{I}$ verifies $\mathcal{R}$, chooses $s_2 \leftarrow\!\!\$\ \mathbb{Z}_p$, computes:

     $$\mathsf{cm}_2 = \mathsf{CM.Com}([\mathsf{id}, s_2]; 0) = g_1^{\mathsf{id}} g_2^{s_2}$$

2. **Aggregation:** Compute combined commitment:

   $$\mathsf{cm} = \mathsf{cm}_1 \cdot \mathsf{cm}_2 = g_1^{\mathsf{id}} g_2^{s_1+s_2} h^r$$

   $\mathcal{I}$ returns $\mathsf{cm}, \mathsf{cm}_2, s_2$ to $\mathcal{U}$

3. **Verification:** $\mathcal{U}$ verifies:

   $$\mathsf{CM.Open}(\mathsf{cm}, (\mathsf{id}, s); r) = \mathsf{CM.Open}(\mathsf{cm}_1 \cdot \mathsf{cm}_2, (\mathsf{pid}, s_1 + s_2); r)$$

4. **Reconstruction:** The shared secret is $s = s_1 + s_2$ where $\mathcal{I}$ only knows $s_2$.

**Security Properties**

- **Hiding:** For $\mathcal{U}$: $\mathsf{rcm}_1$ hides $s_1$ under DDH. For $\mathcal{I}$: $\mathsf{rcm}_2$'s structure binds to $\mathsf{pid}$ without randomness.
- **Binding:** The ZKPoK on $\mathsf{rcm}_1$ prevents $\mathcal{U}$ from choosing $s_1 = 0$. Linear composition ensures:

  $$\Pr[\exists (s_1', a') \neq (s_1, a) : \mathsf{rcm}_1 = g_2^{s_1'} h^{a'}] \leq \mathsf{negl}(\lambda)$$

- **Correctness:** If both parties follow the protocol, $s = s_1 + s_2$ is uniquely determined by:

  $$\log_{g_2}(\mathsf{cm} \cdot g_1^{-\mathsf{id}} h^{-a}) = s_1 + s_2$$

**Application to Anonymous Credentials** This protocol enables:

- **Sybil Resistance:** The $\mathsf{pid}$ binding prevents duplicate issuance
- **Selective Disclosure:** Users can prove $s \neq 0$ without revealing $s_1, s_2$
- **Unlinkability:** Multiple credentials with same $s$ remain unlinkable due to independent $a$

### 13.8 Zero-Knowledge Proof of Non-Zero Secret

To enforce $s_1 \neq 0$ in Protocol 9, we extend the ZKPoK framework with an inverse existence argument.

**Protocol 10: ZKPoK for $s_1 \neq 0$**

**Common Input:** $\mathsf{rcm}_1 = g_2^{s_1} h^a$

**Prover Input:** $s_1 \neq 0, a, t = s_1^{-1} \mod p$

1. **Commitment:**
   - Prover samples $\alpha, \beta, \gamma \leftarrow\$ \mathbb{Z}_p$
   - Computes $T_1 = g_2^{\alpha} h^{\beta}$, $T_2 = g_2^{\gamma}$
   - Sends $(T_1, T_2)$ to verifier
2. **Challenge:** Verifier sends $c \leftarrow\$ \mathbb{Z}_p$
3. **Response:** Prover computes:

$$z_1 = \alpha + ct, \quad z_2 = \beta + cat, \quad z_3 = \gamma + cs_1$$

   Sends $(z_1, z_2, z_3)$
4. **Verification:** Check:

$$g_2^{z_1} h^{z_2} \stackrel{?}{=} T_1 \mathsf{rcm}_1^c \quad \text{and} \quad g_2^{z_3} \stackrel{?}{=} T_2 g_2^c$$

**Security Analysis**

- **Completeness:** Follows from:

$$g_2^{z_1} h^{z_2} = g_2^{\alpha+ct} h^{\beta+cat} = T_1 (g_2^{s_1} h^a)^{ct} = T_1 \mathsf{rcm}_1^c$$
$$g_2^{z_3} = g_2^{\gamma+cs_1} = T_2 g_2^c$$

- **Special Soundness:** From two accepting transcripts $(c, z_1, z_2, z_3)$ and $(c', z_1', z_2', z_3')$:

$$t = \frac{z_1 - z_1'}{c - c'}, \quad s_1 = \frac{z_3 - z_3'}{c - c'}, \quad a = \frac{(z_2 - z_2') - t(z_3 - z_3')}{c - c'}$$

  The existence of $t = s_1^{-1}$ proves $s_1 \neq 0$

- **HVZK:** Simulator $\mathcal{S}$ on input $c$:
  1. Samples $z_1, z_2, z_3 \leftarrow\$ \mathbb{Z}_p$

2. Computes $T_1 = g_2^{z_1} h^{z_2} \mathsf{rcm}_1^{-c}$, $T_2 = g_2^{z_3} g_2^{-c}$
3. Outputs $(T_1, T_2, c, z_1, z_2, z_3)$

Indistinguishable due to uniform sampling in $\mathbb{Z}_p$

**Protocol Integration** Replace Step 1(a) in Protocol 9 with:

- Prove $\mathcal{R}_1' = \{(\mathsf{rcm}_1, (s_1, a, t)) \mid \mathsf{rcm}_1 = g_2^{s_1} h^a \wedge s_1 \cdot t \equiv 1 \mod p\}$
- Issuer verifies both commitment structure and inverse relationship

**Implications**

- **Sybil Resistance:** Prevents $s_1 = 0$ trivial solutions
- **Compatibility:** Maintains $\Sigma$-protocol structure for composition

### 13.9 Equality Proofs Across Multiple Commitments

In many scenarios, $\mathcal{P}$ will want to prove that they have a message consistent across their credentials. For example, a consistent user id $\mathsf{id}$, the proof will attest that the prover is the owner of all credentials. The protocol is presented as a commitment to a single attribute $m_i$ and randomness $r_i$, however the protocol is generalized to any length message vector and the message being in any position, held secure by the commitments position binding property.

$$\mathcal{R}_{\mathsf{eq}} = \left\{ \left( (\mathsf{cm}_1, \mathsf{cm}_2), (m_1, r_1, r_2) \right) \,\middle|\, \mathsf{cm}_1 = g^{m_1} h^{r_1} \wedge \mathsf{cm}_2 = g^{m_1} h^{r_2} \right\}.$$

**Protocol 11: Equality of Secret Exponents between Commitments**

**Common Input:** Commitments $\{\mathsf{cm}_i = g^{m_i} h^{r_i}\}_{i=1}^{k}$

**Prover Input:** $\{m_i, r_i\}$ with $m_1 = \cdots = m_k = m$

**Relation:** $\mathcal{R}_{\mathsf{eq}} = \{(\{\mathsf{cm}_i\}, m, \{r_i\}) \mid \mathsf{cm}_i = g^m h^{r_i} \ \forall i\}$

1. **Commitment:**
   - For each $i$, prover samples $\alpha_i, \rho_i \leftarrow\!\$\ \mathbb{Z}_p$
   - Compute $T_i \leftarrow g^{\alpha_i} h^{\rho_i}$, send $\{T_i\}$ to verifier
2. **Challenge:** Verifier sends $c \leftarrow\!\$\ \mathbb{Z}_p$
3. **Response:**
   - Compute $s \leftarrow \alpha_1 + cm$ and $u_i \leftarrow \rho_i + cr_i$ for all $i$
   - Send $(s, \{u_i\})$
4. **Verification:** For each $i$:

$$g^s h^{u_i} \stackrel{?}{=} T_i \cdot \mathsf{cm}_i^c$$

*Security Analysis:*

- **Soundness:** Extractor obtains $m = \frac{s-s'}{c-c'}$ from two transcripts, proving $m_i = m \ \forall i$

- **ZK:** Simulator chooses $s, \{u_i\} \leftarrow\!\!\$\ \mathbb{Z}_p$, computes $T_i = g^s h^{u_i} \mathsf{cm}_i^{-c}$

*Integration with Previous Protocols:* Combine with Protocol 6 to prove both equality and linear relationships simultaneously:

**Protocol 12: Signature of Knowledge for PS Signatures**

**Common Input:** $(\mathsf{vk}, \mathsf{cm}, \widetilde{\sigma})$ where $\widetilde{\sigma} = (\widetilde{\sigma_1}, \widetilde{\sigma_2})$

**Prover Input:** $(r, \{m_i\}_{i=1}^{\ell})$ such that $\mathsf{cm} = g^r \prod_{i=1}^{\ell} g_i^{m_i}$

1. **Commitment:** Prover samples $\alpha, \{\beta_i\}_{i=1}^{\ell} \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes:
$$T_1 \leftarrow g^{\alpha} \prod_{i=1}^{\ell} g_i^{\beta_i}$$
Send $T_1$ to verifier.
2. **Challenge:** Verifier samples $c \leftarrow\!\!\$\ \mathbb{Z}_p$ and sends to prover.
3. **Response:** Prover computes:
$$s_r \leftarrow \alpha + c \cdot r \quad \text{and} \quad s_i \leftarrow \beta_i + c \cdot m_i \text{ for } i \in [1, \ell]$$
Send $(s_r, \{s_i\}_{i=1}^{\ell})$ to verifier.
4. **Verification:** Verifier checks:
$$g^{s_r} \prod_{i=1}^{\ell} g_i^{s_i} \stackrel{?}{=} T_1 \cdot \mathsf{cm}^c$$
$$e(g, \widetilde{\sigma_2}) \stackrel{?}{=} e(\mathsf{vk} \cdot \mathsf{cm}, \widetilde{\sigma_1})$$

### 13.10 Signature of Knowledge with Rerandomizable PS Signatures

Given the rerandomizable signature scheme

- $\mathsf{CM.Setup}: \mathsf{ck} \leftarrow (g, (g_1, \ldots, g_\ell), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_\ell))$
- $\mathsf{RS.KeyGen}: (\widetilde{\mathsf{sk}}, \mathsf{vk}) \leftarrow (\tilde{g}^x, g^x)$, return $(\widetilde{\mathsf{sk}}, \mathsf{vk}))$
- $\mathsf{RS.Sign}: \widetilde{\sigma} \in \mathbb{G}_2 \leftarrow (\widetilde{\sigma_1}, \widetilde{\sigma_2}) = (\tilde{h}, (\widetilde{\mathsf{sk}} \cdot \widetilde{\mathsf{cm}})^u)$
- $\mathsf{RS.Ver}(\mathsf{vk}, \mathsf{cm}, \widetilde{\sigma}) \to \{0, 1\}$ :

$$e(g, \widetilde{\sigma_2}) = e(\mathsf{vk} \cdot \mathsf{cm}, \widetilde{\sigma_1}) \quad \wedge \quad \pi \leftarrow \mathsf{PoK}\{(r, m_1, \ldots, m_\ell) : \mathsf{cm} = g^r \prod_{i=1}^{\ell} g_i^{m_i}$$

We want to prove the following relation

$$\mathcal{R} = \left\{ (\mathsf{cm}, \widetilde{\sigma}), (r, \{m_i\}) \;\middle|\; e(g, \widetilde{\sigma_2}) = e(\mathsf{vk} \cdot \mathsf{cm}, \widetilde{\sigma_1}) \quad \wedge \quad \mathsf{cm} = g^r \prod_{i=1}^{\ell} g_i^{m_i} \right\}$$

*Security Properties:* This -protocol satisfies:

- **Special Soundness:** Consider two accepting transcripts for the same first message:

$$(T_1, c, s_r, \{s_i\}) \quad \text{and} \quad (T_1, c', s'_r, \{s'_i\})$$

with $c \neq c'$. From the verification equations:

$$g^{s_r} \prod_{i=1}^{\ell} g_i^{s_i} = T_1 \cdot \mathsf{cm}^c \quad \text{and} \quad g^{s'_r} \prod_{i=1}^{\ell} g_i^{s'_i} = T_1 \cdot \mathsf{cm}^{c'}$$

Dividing these equations:

$$g^{s_r - s'_r} \prod_{i=1}^{\ell} g_i^{s_i - s'_i} = \mathsf{cm}^{c-c'}$$

Since $s_r = \alpha + cr$ and $s_i = \beta_i + cm_i$, we can extract:

$$r = \frac{s_r - s'_r}{c - c'} \quad \text{and} \quad m_i = \frac{s_i - s'_i}{c - c'} \text{ for } i \in [1, \ell]$$

- **HVZK:** We construct simulator $\mathcal{S}$ as follows:

1. On input $(\mathsf{cm}, c)$, sample $s_r, \{s_i\} \leftarrow\$ \mathbb{Z}_p$
2. Compute commitment:

$$T_1 \leftarrow g^{s_r} \prod_{i=1}^{\ell} g_i^{s_i} \cdot \mathsf{cm}^{-c}$$

3. Output $(T_1, c, s_r, \{s_i\})$

The simulated transcript is perfectly indistinguishable from real transcripts because:

* In real protocol: $s_r = \alpha + cr$ and $s_i = \beta_i + cm_i$ where $\alpha, \beta_i$ are uniform
* Thus real responses are uniform in $\mathbb{Z}_p$, matching simulator's distribution
* $T_1$ is uniquely determined in both cases by verification equation

## 13.11 Correctness of Keys for protection against Malicious Issuer

Given the commitment key and verification key, we require the issuer to prove in zero knowledge the correctness of their key computation.

- $\mathsf{CM.Setup}$ :$\mathsf{ck} \leftarrow (g, (g_1, \ldots, g_\ell), \tilde{g}, (\tilde{g}_1, \ldots, \tilde{g}_\ell))$
- $\mathsf{RS.KeyGen}$ :$(\mathsf{sk}, \mathsf{vk}) \leftarrow (g^x, \tilde{g}^x)$, return $(\mathsf{sk}, \mathsf{vk})$

We want to prove the following relation:

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

**Protocol 13: Key Validation Protocol**

**Common Input:** $(\mathsf{vk}, \{g_i, \tilde{g}_i\}_{i=1}^{\ell})$

**Prover Input:** $(x, \{y_i\}_{i=1}^{\ell})$ such that $\mathsf{vk} = \tilde{g}^x$ and $g_i = g^{y_i}, \tilde{g}_i = \tilde{g}^{y_i}$

1. **Commitment:** Prover:
   - Samples $\alpha, \{\beta_i\}_{i=1}^{\ell}, \rho \leftarrow\$ \mathbb{Z}_p$
   - Computes commitment to sk: $C_{sk} = g^x h^\rho$
   - Computes $\tilde{T}_1 = \tilde{g}^\alpha$
   - For $i \in [1, \ell] : T_i = g^{\beta_i}, \tilde{T}_i = \tilde{g}^{\beta_i}$

   Send $(C_{sk}, \tilde{T}_1, \{T_i, \tilde{T}_i\}_{i=1}^{\ell})$ to verifier.
2. **Challenge:** Verifier samples $c \leftarrow\$ \mathbb{Z}_p$ and sends to prover.
3. **Response:** Prover computes:
   - $z_x = \alpha + cx$
   - $z_\rho = \rho + cr'$ where $r'$ is randomness used in $C_{sk}$
   - For $i \in [1, \ell] : z_i = \beta_i + cy_i$

   Send $(z_x, z_\rho, \{z_i\}_{i=1}^{\ell})$ to verifier.
4. **Verification:** Verifier checks:

$$\tilde{g}^{z_x} \stackrel{?}{=} \tilde{T}_1 \cdot vk^c$$
$$g^{z_x} h^{z_\rho} \stackrel{?}{=} C_{sk}^c$$
$$\forall i \in [1, \ell] : g^{z_i} \stackrel{?}{=} T_i \cdot g_i^c$$
$$\tilde{g}^{z_i} \stackrel{?}{=} \tilde{T}_i \cdot \tilde{g}_i^c$$

*Security Properties:* This $\Sigma$-protocol satisfies:

- **Special Soundness:** Given two accepting transcripts $(C_{sk}, \tilde{T}_1, \{T_i, \tilde{T}_i\}, c, z_x, z_\rho, \{z_i\})$ and $(C_{sk}, \tilde{T}_1, \{T_i, \tilde{T}_i\}, c', z'_x, z'_\rho, \{z'_i\})$ with $c \neq c'$, we can extract:

$$x = \frac{z_x - z'_x}{c - c'} \quad \text{and} \quad y_i = \frac{z_i - z'_i}{c - c'} \text{ for } i \in [1, \ell]$$

The validity of the extracted values follows from the verification equations:

$$\begin{aligned} \tilde{g}^{z_x} &= \tilde{T}_1 \cdot vk^c & &\Longrightarrow \tilde{g}^{x} = vk \\ g^{z_x} h^{z_\rho} &= C_{sk}^c & &\Longrightarrow g^{x} = C_{sk}/h^{\rho} \\ g^{z_i} &= T_i \cdot g_i^c & &\Longrightarrow g^{y_i} = g_i \\ \tilde{g}^{z_i} &= \tilde{T}_i \cdot \tilde{g}_i^c & &\Longrightarrow \tilde{g}^{y_i} = \tilde{g}_i \end{aligned}$$

- **HVZK:** We construct simulator $\mathcal{S}$ as follows:

1. On input $c$, sample $z_x, z_\rho, \{z_i\} \leftarrow\$ \mathbb{Z}_p$
2. Sample $\rho \leftarrow\$ \mathbb{Z}_p$ and compute $C_{sk} = g^x h^\rho$
3. Compute commitments:

$$\begin{aligned} \tilde{T}_1 &= \tilde{g}^{z_x} \cdot vk^{-c} \\ T_i &= g^{z_i} \cdot g_i^{-c}, \tilde{T}_i = \tilde{g}^{z_i} \cdot \tilde{g}_i^{-c} \end{aligned}$$

4. Output $(C_{sk}, \tilde{T}_1, \{T_i, \tilde{T}_i\}, c, z_x, z_\rho, \{z_i\})$

The simulated transcript is perfectly indistinguishable from real transcripts because:

* In real protocol: $z_x = \alpha + cx$ and $z_i = \beta_i + cy_i$ where $\alpha, \beta_i$ are uniform
* $z_\rho$ is uniform as it masks the secret commitment randomness
* All responses are uniform in $\mathbb{Z}_p$, matching simulator's distribution
* Commitments are uniquely determined by verification equations

## 14 Private Identity System from MIMC-ABC

Our identity system establishes a secure framework for issuing and managing privacy-preserving credentials across multiple authorities while maintaining accountability. The system involves four key entities: users, credential oracles (which verify and attest to user attributes), auditor (who manage revocation), and credential verifiers.

At the core of our system is a master credential issued by a government credential oracle, which serves as a root of trust. This credential contains two crucial committed elements: a secret identifier $s$ that enables secure credential linking, and a committed VRF key $k$ that generates context-specific nullifiers. These nullifiers serve dual purposes: preventing Sybil attacks at credential oracles and enabling efficient revocation. During master credential issuance, the VRF key is verifiably encrypted, the ciphertext is stored in the government system which associates a plaintext user profile to their ciphertext for revocation.

Users can obtain context credentials from various credential oracles by proving possession of a valid, unrevoked master credential and deriving a unique nullifier using their VRF key and the credential context. This design allows credential oracles to restrict issuance to users with trusted government-issued credentials, ensuring their credentials are only issued to verified identities. The system supports expressive verification statements that can combine attributes across multiple credentials. Since master credentials are government-issued and require stringent security checks, verifiers can leverage this trust by incorporating master credential validity, expiration, and revocation checks into their verification statements, inheriting the strong security properties of government-issued credentials. This enables credential oracles to maintain trust by ensuring their credentials become unusable if the underlying government credential is revoked.

The system supports flexible revocation through two mechanisms: targeted revocation of specific credentials via their nullifiers, and complete revocation of all user credentials by recovering their VRF key through the auditors. Government systems can initiate revocation by using plaintext identifiers, with auditors managing the conversion to the appropriate nullifiers. This approach maintains privacy while enabling practical accountability and administration.

We use the Multi Issuer Multi Credential Anonymous Credential system to implement a privacy-preserving digital identity system

### 14.1 Entities

Our identity system involves users, credential issuers, auditors, and credential verifiers.

**User** ($\mathcal{U}$) holds a master credential $Cred_{master}$ and any number of context credentials $Cred_{ctx}$ in their identity wallet. The master credential contains a unique identifier $s$, a VRF key $k$, and additional attributes, and is issued by a government entity. Context credentials are issued by participating organizations like universities or licensing authorities.

**Credential Oracle** ($\mathcal{M}, \mathcal{C}$) verifies user identity and issues digital credentials. The Master Credential Oracle $\mathcal{M}$ operates with keypair $(SK_m, PK_m)$ for issuing "root" credentials, while Context Credential Oracles $\mathcal{C}$ use $(SK_c, PK_c)$ for issuing domain-specific credentials.

**Auditor** ($\mathcal{A}$) consists of a threshold of nodes holding encryption and accumulator keypairs; for simplicity, we refer to both as $(sk_A, pk_A)$. Users encrypt their VRF keys under the auditors' public key, as in key escrow schemes. Auditors can decrypt this key during revocation. The Auditor updates the revocation list.

**Verifier** ($\mathcal{V}$) represents any party wishing to verify a user's credentials.

### 14.2 Data Objects

We now describe the data objects that form our privacy-preserving decentralized identity system. At its core, a Master Credential serves as a root of trust, from which Context Credentials can be derived. During Context Credential issuance, users generate a deterministic nullifier unique to each context using their Master Credential's secrets and the context string, enabling privacy-preserving credential linking.

**Master Credential $\boldsymbol{Cred_m}$** : A master credential is a high-security root credential issued by a government entity containing:

- Identity string $s$: a unique identifier
- VRF key $k$: used to generate context-specific nullifiers
- Context type $ctx$: always set to "master" for master credentials
- Additional attributes $attrs$: including expiry date, date of birth, etc.
- Credential Structure:
  - ∗ Master Commitment $C_m = Com([s, k, ctx, attrs], r)$: A Pedersen commitment to the credential attributes using randomness $r$
  - ∗ Oracle signature $\sigma_m$: A rerandomizable signature over $C_m$, verifiable under $PK_m$

**Master Credential Oracle Data Record:** Following successful master credential issuance, the oracle maintains a record containing:

- Commitment-Signature Pair $(C_m, Cred_m)$:
  - ∗ Master commitment $C_m = Com([s, k, ctx, attrs], r)$: the Pedersen commitment over credential attributes
  - ∗ Oracle Signature $Cred_m$ The signature over commitment $C_m$
- Key Encryption and Proof:
  - ∗ Encrypted VRF Key $CT_k$: the encryption of the user's VRF key, encrypted with the Auditor's public key $Enc_{PK_a}(k)$
  - ∗ Consistency proof $\Pi_{CT}$: The zero-knowledge proof that $CT_k$ encrypts the committed key $k$

**Context Credential $\boldsymbol{Cred_c}$** : A user interacts with the Context Credential oracle to obtain a context-specific credential, which contains:

- Identity string $s$: The user's unique identifier from their master credential
- Nullifier $\tau$: A deterministic value generated from $(s, ctx)$
- Context string $ctx$: A hashed identifier of the credential type (e.g., $dmv$, $universityofsydney$)
- Attribute list $attrs$: Additional credential-specific information such as expiry date
- $\sigma_c$ the rerandomizable signature over $C_c$ from the context credential oracle that proves the user has been issued $Cred_c$ over $C_c$
- Credential Structure:
  - ∗ Context commitment $C_c$: A Pedersen commitment $Com([s, \tau, ctx, attrs], r')$ to the credential attributes using randomness $r'$
  - ∗ Oracle signature $\sigma_c$: A rerandomizable signature over $C_c$, verifiable under $PK_c$

**Context Credential Oracle Data Record:** During credential issuance, the oracle maintains a record of the interaction containing:

- Master Credential Verification:
  - ∗ Randomized credential $Cred'_m$: a rerandomized version of the master credential
  - ∗ Randomized commitment $C'_m$: the corresponding rerandomized commitment
  - ∗ Opening proof $\Pi_{ComOpen}$: Zero-knowledge proof of correct commitment opening
  - ∗ Revocation proof $\Pi_{NonRevoked}$: Zero-knowledge proof that the credential has not been revoked
- Nullifier Components:
  - ∗ Context nullifier $\tau$: The value $VRF(k, ctx)$ derived from the user's committed VRF key and credential context
  - ∗ Derivation proof $\Pi_\tau$: Zero-knowledge proof establishing that
    - · The VRF computation is correct
    - · The key $k$ matches the one committed in $Cred_m$
    - · The context string $ctx$ is correctly incorporated

**Revocation List:**

- Accumulator Structure:
  - ∗ Accumulator value $A$: The current state of the accumulator representing non-revoked credentials
  - ∗ Secret key $sk_A$: The accumulator manager's key for updates
  - ∗ Auxiliary information $aux$: Additional data needed for witness updates
- Revoked Elements:
  - ∗ Master revocations $k$: VRF keys of revoked master credentials
  - ∗ Context revocations $\tau$: Nullifiers of revoked context credentials
  - ∗ Timestamp $t$: Time of revocation
  - ∗ Reason code $rc$: Justification for revocation
- Witness Management:
  - ∗ Non-membership witness $w$: Proof that a credential is not in the revocation set
  - ∗ Update information $upd$: Data for users to update their witnesses after accumulator changes

Notes on Threat/Trust model: threat model has issuer/verifier, trust model has credd oracle, auditor, etc. Keep consistent. Also, state what's out of scope e.g. network, physical, side-channel. domain in arke = context

**Trust Model**

- Credential Oracles: trusted to verify real-world identity before issuing credentials, they aren't trusted for privacy and may be compromised but can't issue credentials without the user participating in their protocol
- Auditors: are trusted to only decrypt user keys for legitimate revocation requests
- Network: communication assumed to be over encrypted channels, any storage is not trusted for credential contents

**Threat Model** We assume the auditor maintaining the revocation cannot be corrupted.

- Malicious Credential Oracle: A malicious credential oracle could "falsely issue attestations and impersonate any user it desires. Fortunately, recent work on authenticating web data has shown privacy-preserving, untrusted and correct credential oracles can be realized in practice [DECO, distefano, etc]. Additionally, we mitigate the threat level by confining each credential oracle to a unique domain." - from Arke.
- Malicious User: attempts to obtain multiple credentials for the same context, tries to forge credentials or share them with others, attempts to link credentials with other credentials not issued to the same master secret key
- Malicious Issuer: attempts to link multiple showing, collude with issuers to deanonymize users, stores presentation proofs to track users
- Malicious Verifier: issue credentials without proper verification, attempts to track credential usage, colludes with issuers or other verifiers

## 14.3 Syntax

Sam: Sam to update this to include master and context cred and revocation

Syntax of Anonymous Identity System with Sybil Resistance and Revocation

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathcal{UL}, \mathcal{RL})$ Takes security parameter $\lambda$ in unary, outputs public parameters $\mathsf{pp}$, empty user list $\mathcal{UL}$ and revocation list $\mathcal{RL}$.
- $\mathsf{OrgKeygen}(\mathsf{pp}, n)(\mathsf{osk}, \mathsf{opk})$: is a probabilistic algorithm that takes public parameters $\mathsf{pp}$ and $n$ the upper bound of credential attributes. Outputs organisations keypair $(\mathsf{osk}, \mathsf{opk})$
- $\mathsf{UserKeygen}(1^\lambda)(\mathsf{usk})$ is a probabilistic algorithm. Outputs user secret key $\mathsf{usk}$ consisting of PRF key $k$ and identity string $s$
- $(\mathsf{ObtainMaster}(usk, \mathbf{m}, aux), \mathsf{IssueMaster}(osk, cm, aux)) \to cred$ : An interactive protocol. *Obtain* is a probabilistic alrogithm run by a user, inputs secret key, credential attribute vector $\mathbf{m}$ and auxiliary info. *Issue* is a probabilistic algorithm run by an issuing organization that takes a commitment $cm$, issuers secret key $osk$, and auxiliary info. Outputs a credential $cred$ binding $cm$ to the issuer signature.
- $(\mathsf{ObtainContext}(usk, \mathbf{m}, aux), \mathsf{IssueContext}(osk, cm, aux)) \to cred$ : An interactive protocol. *Obtain* is a probabilistic alrogithm run by a user, inputs secret key, credential attribute vector $\mathbf{m}$ and auxiliary info. *Issue* is a probabilistic algorithm run by an issuing organization that takes a commitment $cm$, issuers secret key $osk$, and auxiliary info. Outputs a credential $cred$ binding $cm$ to the issuer signature.
- $(\mathsf{Show}(usk, cred, \phi), \mathsf{Verify}(cred', cm, \pi)) \to \{0, 1\}$ An interactive protocol. *Show* is a probabilistic algorithm run by a Prover. Takes secret key, credential, and predicate statement

$\phi$. $Verify$ is a deterministic algorithm run by a verifier, takes a randomized credential and commitment $cred', cm'$ and proof $\pi$. Otuputs 1 if verification succeeds, otherwise 0.

- $\mathsf{Revoke}(\mathcal{RL}, k') \to \mathcal{RL}'$ revoke is a deterministic algorithm, updates revocation list with revoked key $k'$

## 14.4 Security Model

- **Sybil resistance**: For any given context, no probabilistic polynomial time adversary can obtain more than 1 valid credential with non-negligible probability
- **Revocability**: For any given context, no probabilistic polynomial time adversary use a revoked credential

### Sybil Resistance

**Revocation** When $ra$ needs to revoke a user's credential/s (due to user request or credential provider request), $ra$ finds $escrow$ based on the user's $pid$, recall $ra$ has a user list $ul = (pid, escrow)$ and requests the auditor $audit$ to decrypt $s \leftarrow tpkdec_{ask}(escrow)$. $audit$ computes the nullifiers to add to the revocation accumulator. $nullif_{rcd} \leftarrow PRF_s(pid)$ and for each context credential to revoke, $nullif_{ctxid} \leftarrow PRF_s(ctxid)$. $audit$ updates the accumulator $acc' \leftarrow Acc.Add(acc)$ If the registration credential requires revocation, $audit$ can compute each $nullif \leftarrow PRF_s(ctxid); \forall; ctxid; \in; ctxl$ and add $(nullif, timestamp, reason)$ to $rl$. For record-keeping, $ra$ stores Revocation Information $ri = (nullif, timestamp, reason)$ allowing $ra$ to track which credentials are revoked and why, $nullif$ in $rl$ ensures revoked credentials can't be verified. During credential verification, verifiers check if a credential's nullifier appears in $rl$, if present, the verification fails.

Table 2: Multi-Credential Verification Performance

| **Scheme** | **5 Creds** | **10 Creds** | **20 Creds** |
|---|---|---|---|
| BBS+ [?] | xxx ms | xxx ms | xxx ms |
| Standard PS [?] | xxx ms | xxx ms | xxx ms |
| PS-UTT (Ours) | xxx ms | xxx ms | xxx ms |

*All measurements averaged over 100 runs with 10 attributes per credential*

Table 3: Detailed Performance Analysis of Multi-Credential Systems

| | | **Number of Credentials** | | |
|---|---|---|---|---|
| a **Scheme** | **Metric** | **5** | **10** | **20** |
| BBS+ | Prover Time (ms) | xxx | xxx | xxx |
| | Verifier Time (ms) | xxx | xxx | xxx |
| | Proof Size (bytes) | xxx | xxx | xxx |
| Standard PS | Prover Time (ms) | xxx | xxx | xxx |
| | Verifier Time (ms) | xxx | xxx | xxx |
| | Proof Size (bytes) | xxx | xxx | xxx |
| PS-UTT (Ours) | Prover Time (ms) | xxx | xxx | xxx |
| | Verifier Time (ms) | xxx | xxx | xxx |
| | Proof Size (bytes) | xxx | xxx | xxx |

*All measurements performed on [CPU specs] with 10 attributes per credential, averaged over 100 runs*

## 15 Performance Evaluation

What is the takeaway message from the evaluation?

- For non-private system, we enable privacy with little overhead. Our building block sigma protocol for private sybil resistance adds negligible overhead.
- For private system, but better efficiency, we have a SOTA paper TACT/S3ID (in the comparison table). Their paper does multi-attribute/multi-issuer credentials (they issue 1 credential per attribute), but their benchmarks don't show the complexity in verifying credentials together, or proving statements about their credential which they say would have non-negligible impact and theirs is lower bound. For us, by using simpler and well-known construction, we are more efficient (need to test this but I think so due to their construction) and better functionality.

## 16 Efficient Multi-Show Anonymous Credentials via Algebraic Decoupling

### 16.1 Introduction

Privacy-preserving authentication systems require cryptographic primitives that allow users to prove certified attributes (e.g., citizenship, qualifications) without revealing additional information or enabling linkage across sessions. Anonymous credential schemes address this by encoding attributes into cryptographically randomized signatures, enabling users to generate unlinkable "showings" of credentials through zero-knowledge proofs (ZKPs). A critical challenge lies in balancing three properties: (1) multi-show unlinkability (proving attributes without correlating sessions), (2) selective disclosure (revealing subsets of attributes), and (3) practical efficiency for real-world deployment.

The problems in accountable privacy were first introduced by Chaum [Cha81, Cha85] and later formalized as the *CL framework* [LRSW00, CL01], establishing the paradigm of encoding credentials as signatures over committed attributes. While early signature schemes lacked the algebraic structure to support efficient ZKPs and signature randomization, the advent of pairing-based cryptography [BLS01, BBS04] coupled with commitment schemes and zero-knowledge proof systems enabled homomorphic rerandomization of credentials—allowing users to transform signatures into fresh, unlinkable versions while preserving validity.

### 16.2 Evolution of Anonymous Credential Schemes

The progression of anonymous credential schemes reflects a sustained effort to simplify the mathematical structures underlying zero-knowledge proofs:

*CL Signatures [CL03, CL04]* Built on early pairing-based constructions, CL credentials introduced homomorphic rerandomization but required linear-in-attributes pairing operations during proofs, limiting scalability for complex credentials.

*BBS+ Signatures [ASM06]* Achieved constant-size credentials by intertwining message commitments with signatures. While [CDL16] improved computational complexity, The signature leverages the q-SDH assumption [BB08], requiring a mathematical structure that which intertwines the message and secret key $A^{\frac{1}{x+e}}$ which enables zkp statements but complicates proofs

*BBS+ Signatures [ASM06]* Achieved constant-size credentials through a structure where signatures take the form $A = (gh_0^s \prod_{i=1}^n h_i^{m_i})^{\frac{1}{e+x}}$. While [CDL16] improved computational complexity, the signature structure necessitates complex blinding operations with inverse relationships ($r_3 = \frac{1}{r_1}$), cross-terms ($s' = s - r_2 \cdot r_3$), and complex equality checks during proof generation, e.g.

$$g = ((gh_0^s \prod_1^n h_i^{-m_i})^{r_1} \cdot h_0^{-r_2})^{r_3} h_0^{-s'} \prod_{i=1}^n h_i^{-m_i}$$

This leads to a zero-knowledge proof relation involving multiple auxiliary variables and non-linear equations, complicating the implementation and composition of proofs across multiple credentials.

*PS and PS_ UTT Signatures [PS16, TBA$^+$22]* Introduced separation between signature components and message commitments. The UTT variant further simplified rerandomization by decoupling attribute commitments from the core signature structure, enabling linear ZKP relations over Pedersen commitments. PS based signature rely on the LRSW assumption [LRSW00] rather than BBS+ q-SDH which enables simpler algebraic structures, which we show now.

### 16.3 Cryptographic Foundations: q-SDH vs LRSW

**q-SDH**

**Definition 29 (q-Strong Diffie-Hellman Assumption).** *For any PPT adversary $\mathcal{A}$ and positive integer $q = \mathsf{poly}(n)(\lambda)$, we say the q-SDH assumption holds if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\begin{array}{ll}\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \tilde{g}) \leftarrow\!\!\$\ \mathsf{BGGen}(1^\lambda) & \\ s \leftarrow\!\!\$\ \mathbb{Z}_p & : e(h, \tilde{g}^s \cdot \tilde{g}^c) = e(g, \tilde{g}) \\ (c, h) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{BG}, g, \tilde{g}, \tilde{g}^s, \dots, \tilde{g}^{s^q}) & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

*where $c \in \mathbb{Z}_p \setminus \{-s\}$.*

The choice between these shapes the proof complexity:

*BBS+ and q-SDH Limitations* The $q$-Strong Diffie-Hellman (q-SDH) assumption [BB08] requires signatures of the form $A = (g_0 g_1^s \prod h_i^{m_i})^{\frac{1}{e+x}}$. This structure:

- Demands *inverse exponentiation* ($1/(e+x)$), creating non-linear relationships between secrets
- Forces proofs to handle *cross-term cancellation* (e.g., $A_1^e = g_1^{\delta_1} g_2^{\delta_2}$)
- Requires pairing inversions ($e(A_2, \hat{h}_0)^{-e}$) to verify credential validity

*PS_UTT and LRSW Advantages* The LRSW assumption [?] underpinning PS_UTT enables signatures with *linear* algebraic structure:

- Signatures take form $\sigma = (h^u, (X \cdot \prod g_i^{m_i} \cdot h^r)^u)$ with *no inverse operations*
- Randomization factors ($u_\Delta, r_\Delta$) modify signatures additively rather than multiplicatively
- Verification equations remain bilinear pairings without cross-term cancellation

*Assumption-Driven Efficiency* The LRSW assumption enables three key efficiency properties, each building on the previous:

- **Linear Exponent Relations**: Messages appear directly in the exponent as $\prod g_i^{m_i}$, avoiding the nested fraction structure of q-SDH
- **Independent Randomization**: The separation of $u_\Delta$ and $r_\Delta$ creates a natural two-layer randomization strategy, unlike BBS+'s entangled randomized

Sam(does it change how SNARKS verify these signatures, multiplicative inverse in circuits?)

## 16.4 The PS_UTT Advantage: Algebraic Decoupling

PS_UTT's key innovation lies in its separation of the signature's algebraic structure from the attribute commitment layer. Consider the ZKP relation for proving knowledge of PS_UTT-signed attributes:

$$\pi \leftarrow \mathsf{PoK}\{(m_1, \dots, m_\ell, u_\Delta, r_\Delta) : e(\sigma_2, \hat{h}) = e(\sigma_1, \widehat{X} \cdot \widehat{\mathsf{cm}}) \wedge e(\mathsf{cm}, \hat{h}) = e(h, \widehat{\mathsf{cm}}) \wedge \mathsf{cm} = h^{r + r_\Delta} \prod_{i=1}^{\ell} g_i^{m_i}\} \tag{7}$$

Here, the prover need only linearly combine public parameters $(g_i, h)$ with secrets $(m_i, r_\Delta)$, leveraging the homomorphism of Pedersen commitments. This contrasts sharply with BBS+'s non-linear relation:

$$\pi \leftarrow \mathsf{SPK}\{(m_1, \dots, m_\ell), e, r_2, r_3, s') : \frac{\bar{A}}{d} = A'^{-e} \cdot h_0^{r_2} \wedge g = d^{r_3} h_0^{-s'} \prod_{i=1}^{n} h_i^{-m_i}\} \tag{8}$$

### 16.5 Technical Advantages

*Simplified Zero-Knowledge Proofs* By decoupling randomization $(u_\Delta, r_\Delta)$ from the core signature verification equation, PS_UTT eliminates the need for complex cancellation of cross-terms during proofs. This reduces both the number of group operations and the interactivity required in ZKP composition.

*Efficient Multi-Credential Proofs* The separation of signature randomization from attribute proving enables more efficient composition of proofs across multiple credentials. Where BBS+ requires handling interleaved inverse operations and auxiliary variables, PS_UTT's linear structure allows direct combination of attribute proofs through standard Pedersen commitment arithmetic.

### 16.6 Practical Impact

Our analysis demonstrates significant practical advantages:

SamDo some testing on this and hopefully it will hold!

- **Proof Size Reduction**: PS_UTT reduces proof sizes by (X Percent) compared to BBS+ in comparable settings
- **Verification Performance**: Benchmarks show PS_UTT proofs are X TIMES FASTER 2.1 faster to generate and verify than BBS+ at the 128-bit security level.
- **Implementation Simplicity**: The linear structure of PS_UTT proofs simplifies implementation and reduces the risk of subtle errors in proof composition.

### 16.7 Conclusion

PS_UTT represents a significant advancement in anonymous credential systems through its elegant separation of signature, commitment, and randomization components. This modularity enables simpler zero-knowledge proofs, more efficient verification, and natural compatibility with modern proof systems—making it particularly suitable for resource-constrained environments and complex multi-credential scenarios.

### 16.8 PS [PS16]

The signature scheme for signing information-theoretically hidden messages consists of the following algorithms:

- **KeyGen**$(1^\lambda)$:
  - $*$ Generate bilinear group: $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g}) \leftarrow\!\$\ \mathsf{BGGen}(1^\lambda)$
  - $*$ Sample $(x, y_1, \ldots, y_r) \leftarrow\!\$\ (\mathbb{Z}_p^*)^{\ell+1}$
  - $*$ Compute $(X, Y_1, \ldots, Y_r) \leftarrow (g^x, g^{y_1}, \ldots, g^{y_r})$ and $(\hat{X}, \hat{Y}_1, \ldots, \hat{Y}_r) \leftarrow (\hat{g}^x, \hat{g}^{y_1}, \ldots, \hat{g}^{y_r})$. Set $\mathsf{sk} \leftarrow X$ and $\mathsf{pk} \leftarrow (g, Y_1, \ldots, Y_r, \hat{g}, \hat{g}^{y_1}, \ldots, \hat{g}^{y_r})$
- **BlindSign Protocol**:
  - $*$ **Prover** (input: $m_1, \ldots, m_\ell$): samples $t \leftarrow\!\$\ \mathbb{Z}_p^*$, computes $\mathsf{cm} \leftarrow g^t \prod_{i=1}^r Y_i^{m_i}$. Sends $\mathsf{cm}$ with a proof of knowledge to Signer

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

  - $*$ **Signer** proves $\pi$, on success, samples $u \leftarrow\!\$\ \mathbb{Z}_p^*$, computes $\sigma' \leftarrow (g^u, (X\mathsf{cm})^u)$, returns to Prover

  * **User** unblinds by $\sigma \leftarrow (\sigma_1', \frac{\sigma_2'}{\sigma_1{'}t})$

- **Verify PoK**:

  * **Prover:** Randomizes $\sigma$, samples $r, t \leftarrow\!\!\$\ \mathbb{Z}_p^*$, computes $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$. Sends $\sigma' = (\sigma_1, \sigma_2)$ to $\mathcal{V}$ with a PoK

$$\pi \leftarrow \mathsf{PoK}\{(m_1, \dots, m_\ell, t) : e(\sigma_2', \hat{g}) = e(\sigma_1', \hat{X}) \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{m_j} \cdot e(\sigma', \hat{g})^t\}$$

**Protocol 14: PS Signature: Zero-Knowledge Proof of Knowledge of Blind Signature**

**Relation $\mathcal{R}$:** Prove knowledge of messages and randomness in a blind signature:

$$\pi \leftarrow \mathsf{SPK}\{(m_1, \dots, m_\ell, t) : e(\sigma_2', \hat{g}) = e(\sigma_1', \hat{X}) \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{m_j} \cdot e(\sigma_1', \hat{g})^t\}$$

**Common Input:** $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g})$, $pk = (g, Y_1, \dots, Y_r, \hat{g}, \hat{X}, \widehat{Y}_1, \dots, \widehat{Y}_r)$,

**Private Input:** $\mathcal{P}$ has $(m_1, \dots, m_\ell, t)$ where $\sigma = (\sigma_1, \sigma_2)$, $\sigma_1 = g^u, \sigma_2 = (g^x \cdot \mathsf{cm})^u$ and $\mathsf{cm} = g^t \prod_{i=1}^r Y_i^{m_i}$

1. $\mathcal{P}$: randomizes $\sigma$, samples $r, t \leftarrow\!\!\$\ \mathbb{Z}_p^*$, computes $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$
   - Samples Schnorr blinding factors from $\mathbb{Z}_p^*$

$$\tilde{x} \text{ for } x \in \{\ t, r, m_1, \dots, m_\ell\}$$

   - Compute Schnorr commitment:

$$T \leftarrow e(\sigma_1', \hat{X}) \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{\tilde{m}_j} \cdot e(\sigma_1', \hat{g})^{\tilde{t}}$$

   - Send $T$ to $\mathcal{V}$
2. $\mathcal{V}$ sends challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p^*$
3. $\mathcal{P}$ computes Schnorr responses:

$$z_x = \tilde{x} + c \cdot x \text{ for } x \in \{t, r, m_1, \dots, m_\ell\}$$

   $\mathcal{P}$ sends $\mathcal{V}$ $(\sigma_1', \sigma_2' \in \mathbb{G}_1, T \in \mathbb{G}_T, z_{x_j} \in \mathbb{F}_p^{\ell+2}$
4. $\mathcal{V}$ checks:

$$e(\sigma_2', \hat{g})^c \cdot T \stackrel{?}{=} e(\sigma_1', \hat{X})^c \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{z_{m_j}} \cdot e(\sigma_1', \hat{g})^{z_t}$$

**Notation:**

- $F_p-, F_p\times, F_p+$: Field inversion, multiply, add $\in \mathbb{Z}_p$
- $E_{\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T}$, $M_{\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T}$: Exponentiation / Multiplication in respective groups
- $P$: Pairing Operation
- $\ell$: Number of messages

**PS16 Complexity Analysis [PS16]**

- **Prover's Initial Operations (Signature Randomization)**
  - $(\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r)$ $3E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1}$
- **Prover Schnorr Operations**
  - Sample $\ell + 2$ field elements $(t, r, m_1, \ldots, m_\ell)$ (negligible)
  - Compute $T \leftarrow e(\sigma_1', \hat{X}) \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{\tilde{m}_j} \cdot e(\sigma_1', \hat{g})^{\tilde{t}}$ $(\ell+2)P + (\ell+1)E_{\mathbb{G}_T} + (\ell+1)M_{\mathbb{G}_T}$
- **Proof Size / Data Sent**
  - $(\sigma_1', \sigma_2', T)$ $2\mathbb{G}_1$, $1\mathbb{G}_T$ elements
  - $z_x$ for $x \in \{t, r, m_1, \ldots, m_\ell\}$ $(\ell + 2)\mathbb{Z}_p^*$ elements
- **Verifier Operations**
  - Equation verification: $e(\sigma_2', \hat{g}) \overset{?}{=} e(\sigma_1', \hat{X}) \cdot \prod_{j=1}^{\ell} e(\sigma_1', \widehat{Y}_j)^{z_{m_j}} \cdot e(\sigma_1', \hat{g})^{z_t}$ $(\ell + 3)P + (\ell + 1)E_{\mathbb{G}_T} + (\ell + 2)M_{\mathbb{G}_T}$

### 16.9 PS22 [TBA+22]

- **PS.Rerand**$(\mathsf{pk}, \sigma = (\sigma_1, \sigma_2), \mathsf{cm} = (\mathsf{cm}, \widetilde{\mathsf{cm}})) \to (\sigma', \mathsf{cm}', r_\Delta, u_\Delta)$
  - $*$ $\mathcal{P}$ samples $r_\Delta, u_\Delta \leftarrow\!\$\, \mathbb{Z}_p$
  - $*$ Rerandomizes the signature $\sigma' = (\sigma_1', \sigma_2') \leftarrow (\sigma_1^{u_\Delta}, (\sigma_2 \cdot \sigma_1^{r_\Delta})^{u_\Delta})$
  - $*$ Randomizes the commitment by computing $g^{r_\Delta}, \hat{g}^{r_\Delta}$, then $(\mathsf{cm}', \widehat{\mathsf{cm}'}) \leftarrow (\mathsf{cm} \cdot h^{r_\Delta}, \widehat{\mathsf{cm}} \cdot \hat{h}^{r_\Delta})$
- **PoK**$(\mathsf{pk}, \sigma = (\sigma_1, \sigma_2), \mathsf{cm}, \widehat{\mathsf{cm}}, \vec{m}, r_\Delta, u_\Delta)$ $\mathcal{P}$ sends rerandomized signature and commitment $\sigma', \mathsf{cm}'$ and $\pi$

$$\pi \leftarrow \mathsf{PoK}\{(m_1, \dots, m_\ell, u + u_\Delta, r + r_\Delta) :$$

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

notes, recall

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

$$\mathsf{cm}' = g^{r + r_\Delta} \prod_{i=1}^{\ell} g_i^{m_i}$$

Changing to this

$$\mathsf{cm}' = \tilde{g}^{r_\Delta} \cdot \widetilde{\mathsf{cm}} \text{ without sending } \widetilde{\mathsf{cm}}$$

but using its bases $g_1, \dots, g_\ell$

$$\sigma_1' = \sigma_1^{u_\Delta} = h^{u_\Delta} = g^{u \cdot u_\Delta}$$

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

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

**Protocol 15: Zero-Knowledge Proof of Knowledge for PS Signature over Commitments**

**Relation $\mathcal{R}$**

$$\pi \leftarrow \mathsf{PoK}\{(m_1, \ldots, m_\ell, u + u_\Delta, r + r_\Delta) :$$

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

**Common Input:** $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e)$, $\mathsf{pk} = (\mathsf{vk} = \tilde{X}, \mathsf{ck} = g_1, \ldots, g_\ell, g)$, a rerandomized signature $\sigma' = (\sigma_1', \sigma_2')$, and rerandomized commitment $\mathsf{cm}' = (\mathsf{cm} \cdot g^{r_\Delta}, \widetilde{\mathsf{cm}}' \cdot \tilde{g}^{r_\Delta})$

**Private Input:** $\mathcal{P}$ has messages $(m_1, \ldots, m_\ell)$, commitment randomness $r' = r + r_\Delta$, signature randomness $u' = u + u_\Delta$

1. $\mathcal{P}$:
   - Sample Schnorr blinding factors from $\mathbb{Z}_p$:

     $$\tilde{x} \text{ for } x \in \{m_1, \ldots, m_\ell, r', u'\}$$

   - Compute Schnorr commitments:

     $$T_1 \leftarrow g^{\tilde{r'}} \prod_{i=1}^{\ell} g_i^{\tilde{m_i}} \qquad T_2 \leftarrow e(\sigma_1'^{\tilde{u'}}, \mathsf{vk} \cdot \widehat{\mathsf{cm}'})$$

   - Send $(T_1 \in \mathbb{G}_1, T_2 \in \mathbb{G}_T)$ to $\mathcal{V}$
2. $\mathcal{V}$ sends challenge $c \leftarrow\$ \mathbb{Z}_p$
3. $\mathcal{P}$ computes Schnorr responses:

   $$z_x = \tilde{x} + c \cdot x \text{ for } x \in \{m_1, \ldots, m_\ell, u', r'\}$$

4. $\mathcal{V}$ checks:

   $$\mathsf{cm}'^c \cdot T_1 \stackrel{?}{=} g^{z_{r'}} \prod_{i=1}^{\ell} g_i^{z_{m_i}} \quad \wedge \quad e(\mathsf{cm}', \tilde{g}) \stackrel{?}{=} e(g, \widetilde{\mathsf{cm}}')$$

   $$e(\sigma_2', \tilde{g})^c \cdot T_2 \stackrel{?}{=} e(\sigma_1'^{z_{u'}}, \mathsf{vk} \cdot \widetilde{\mathsf{cm}}')$$

**PS UTT Complexity Analysis [TBA+22]**

- **Prover's Initial Operations (Signature Rerandomization)**
  - $(\sigma_1^{u_\Delta}, (\sigma_2 \cdot \sigma_1^{r_\Delta})^{u_\Delta})$ $3E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1}$
  - $(\mathsf{cm} \cdot h^{r_\Delta}, \widehat{\mathsf{cm}} \cdot \hat{h}^{r_\Delta})$ $1E_{\mathbb{G}_2} + 1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1} + 1M_{\mathbb{G}_2}$
- **Prover Schnorr Operations**
  - Sample $\ell + 2$ field elements $(\tilde{m_1}, \ldots, \tilde{m_\ell}, \tilde{r'}, \tilde{u'})$ (negligible)
  - Compute commitment $T_1 \leftarrow h^{\tilde{r'}} \prod_{i=1}^{\ell} g_i^{\tilde{m_i}}$ $(\ell + 1)E_{\mathbb{G}_1} + \ell M_{\mathbb{G}_1}$
  - Compute commitment $T_2 \leftarrow e(\sigma_1^{\tilde{u'}}, \widehat{X} \cdot \widehat{\mathsf{cm}'})$ $1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_2} + 1P$
- **Proof Size / Data Sent**
  - $(\sigma_1', \sigma_2', \mathsf{cm}', \widehat{\mathsf{cm}'}, T_1, T_2)$ $5\mathbb{G}_1$, $1\mathbb{G}_2$ elements
  - $z_x$ for $x \in \{m_1, \ldots, m_\ell, r', u'\}$ $(\ell + 2)\mathbb{Z}_p$ elements
- **Verifier Operations**
  - First equation check: $\mathsf{cm}'^c \cdot T_1 \overset{?}{=} h^{z_{r'}} \prod_{i=1}^{\ell} g_i^{z_{m_i}}$ $(\ell + 2)E_{\mathbb{G}_1} + (\ell + 2)M_{\mathbb{G}_1}$
  - Second equation check: $e(\mathsf{cm}, \hat{h}) \overset{?}{=} e(h, \widehat{\mathsf{cm}})$ $2P$
  - Third equation check: $e(\sigma_2, \hat{h})^c \cdot T_2 \overset{?}{=} e(\sigma_1^{z_{u'}}, \widehat{X} \cdot \widehat{\mathsf{cm}'})$ $1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_2} + 1E_{\mathbb{G}_T} + 2P + 2M_{\mathbb{G}_T}$

Fig. 7: Complexity comparison between standard PS UTT and optimized variant. The optimized version uses Multi-Scalar Multiplication to reduce Schnorr proof from $\mathcal{O}(\ell)$ individual exponentiation to $\mathcal{O}(log\ell)$ effective operations using MSM techniques. Verifier pairing checks are re-arranged with Miller Loop + Final Exponentiation techniques with Pairing Inversion. Miller Loop batching (MLoop) and final exponentiation sharing (F.Exp), reducing the pairing cost from 4P to approximately 2.5P. MSM(n) denotes an n-point multi-scalar multiplication with complexity O(n/log n) using window methods

| Protocol | Show (Prover) | Verify (Verifier) | Data Sent |
|---|---|---|---|
| PS UTT | **Randomize:**<br>• $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$<br>• $1E_{\mathbb{G}_2} + 1M_{\mathbb{G}_2}$<br>**Schnorr Proof:**<br>• $(\ell+1)E_{\mathbb{G}_1} + \ell M_{\mathbb{G}_1}$<br>• $1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_2} + 1P$ | • $(\ell+3)E_{\mathbb{G}_1} + (\ell+2)M_{\mathbb{G}_1} + 1M_{\mathbb{G}_2}$<br>• $1E_{\mathbb{G}_T} + 2M_{\mathbb{G}_T}$<br>• $4P$ | • $5\mathbb{G}_1$<br>• $1\mathbb{G}_2$<br>• $(\ell+2)\mathbb{Z}_p$ |
| PS UTT Optimized | **Randomize:**<br>• $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$<br>• $1E_{\mathbb{G}_2} + 1M_{\mathbb{G}_2}$<br>**Schnorr Proof:**<br>• ~~$(\ell+1)E_{\mathbb{G}_1} + \ell M_{\mathbb{G}_1}$~~<br>• $MSM(\ell+1)\mathbb{G}_1$<br>• $1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_2} + 1P$ | • ~~$(\ell+3)E_{\mathbb{G}_1} + (\ell+2)M_{\mathbb{G}_1} + 1M_{\mathbb{G}_2}$~~<br>• $MSM(\ell+3) + 1M_{\mathbb{G}_2}$<br>• $1E_{\mathbb{G}_T} + 2M_{\mathbb{G}_T}$<br>• ~~$4P$~~<br>• 4MLoop$\mathbb{G}_T$ + 1F.Exp$\mathbb{G}_T$ | • $5\mathbb{G}_1$<br>• $1\mathbb{G}_2$<br>• $(\ell+2)\mathbb{Z}_p$ |

| **Scheme** | **Show** | | **Verify** | | |
|---|---|---|---|---|---|
| | Credential | PoK | Credential | PoK | Data Sent |
| PSUTT | $(2\ell+3)E_{\mathbb{G}_1}$ | $(2\ell+1)E_{\mathbb{G}_1}$ | $(3\ell+2)P$ | $(\ell+2)E_{\mathbb{G}_1}$ | $(2\ell+3)\lvert\mathbb{G}_1\rvert + \lvert PK\rvert$ |
| PSUTT+LVS | $3E_{\mathbb{G}_2} + E_{\mathbb{G}_1}$ | $MSM(\ell+1)_{\mathbb{G}_1} + E_{\mathbb{G}_1}$ | $2M\text{Loop}\mathbb{G}_T + E_{\mathbb{G}_T}$ | $E_{\mathbb{G}_1} + MSM(\ell+1)_{\mathbb{G}_1}$ | $2\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{G}_2\rvert + (\ell+1)\lvert\mathbb{Z}_p\rvert$ |

Table 4: Comprehensive comparison of anonymous credential schemes. $P$ denotes pairing operation, $E_{\mathbb{G}_1}$, $E_{\mathbb{G}_2}$ and $E_{\mathbb{G}_T}$ denote exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and target group $\mathbb{G}_T$ respectively, $\ell$ denotes number of attributes, $|\mathbb{G}_1|$ and $|\mathbb{G}_2|$ denote group element sizes, and $|\mathbb{Z}_p|$ denotes field element size.

## 16.10 BBS+ 2006

Original BBS+ signature

**Protocol for Signature Proof of Knowledge** asd

**Protocol 16: [ASM06] Zero-Knowledge Proof of Knowledge of BBS+**

**Relation $\mathcal{R}$**

$$\pi \leftarrow \mathsf{SPK}\{(r_1, r_2, e, \delta_1, \delta_2, s, m_1, \ldots, m_\ell)) : \quad A_1 = g_1^{r_1} g_2^{r_2} \quad \wedge \quad A_1^e = g_1^{\delta_1} g_2^{\delta_2} \quad \wedge$$

$$\frac{e(A_2, \widehat{w})}{e(g_2, \widehat{h_0})} \quad = \quad e(g_2, \widehat{w})^{-e} e(g_2, \widehat{w})^{r_1} e(g_1, \widehat{h_0})^{e} (g_2, \widehat{h_0})^{m_1} \cdots e(g_{L+1}, \widehat{h_0})^{m_L}$$

**Common Input:** $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_0, \widehat{h_0})$, $pk = (\hat{w} = \hat{h_0}^{\gamma}, g_0, \ldots, g_L)$

**Private Input:** $\mathcal{P}$ has BBS+ signature $(A, e, s)$ where $A = (g_0 g_1^s \prod_{i=1}^L g_{i+1}^{m_i})^{\frac{1}{e+\gamma}}$ and $\delta_1 = r_1 e$ and $\delta_2 = r_2 e$

1. $\mathcal{P}$ samples random $r_1, r_2 \leftarrow\!\$\ \mathbb{Z}_p^*$, set $\delta_1 = r_1 e$ and $\delta_2 = r_2 e$
   - Compute blinded signature: $A_1 = g_1^{r_1} g_2^{r_2} \quad \wedge \quad A_1^e = g_1^{\delta_1} g_2^{\delta_2} \quad \wedge \quad A_2 = A g_2^{r_1}$
   - sample Schnorr blinding factors from $\mathbb{Z}_p^*$

     $$\tilde{x} \text{ for } x \in \{r_1, r_2, e, \delta_1, \delta_2, s, m_1, \ldots, m_L\}$$

   - compute Schnorr Commitments:

     $$T_1 \leftarrow g_1^{\tilde{r_1}} g_2^{\tilde{r_2}} \qquad T_2 \leftarrow g_1^{\tilde{\delta_1}} g_2^{\tilde{\delta_2}}$$

     $$T_3 \leftarrow e(A_2, \hat{h_0})^{\tilde{-e}} \cdot e(g_2, \hat{w})^{\tilde{r_1}} \cdot e(g_2, \hat{h_0})^{\tilde{\delta_1}} \cdot e(g_1, \hat{h_0})^{\tilde{s}} \cdot e(g_2, \hat{h_0})^{\tilde{m_1}} \cdots e(g_{L+1}, \hat{h_0})^{\tilde{m_L}}$$

   - Send $(A_1, A_1^e, A_2, T_1, T_2, T_3)$ to $\mathcal{V}$
2. $\mathcal{V}$ sends challenge $c \leftarrow\!\$\ \mathbb{Z}_p^*$
3. $\mathcal{P}$ computes Schnorr responses:

   $$z_x = \tilde{x} + c \cdot x \text{ for } x \in \{r_1, r_2, e, \delta_1, \delta_2, e, s, m_1, \ldots, m_\ell)\}$$

4. $\mathcal{V}$ checks

   $$A_1^c \cdot T_1 \overset{?}{=} g_1^{z_{r_1}} g_2^{z_{r_2}} \quad \wedge \quad (A_1^e)^c \cdot T_2 = g_1^{z_{\delta_1}} g_2^{z_{\delta_2}} \quad \wedge$$

   $$\frac{e(A_2, \widehat{w})}{e(g_2, \widehat{h_0})} \overset{?}{=} e(A_2, \hat{h_0})^{z_e} \cdot e(g_2, \hat{w})^{z_{r_1}} \cdot e(g_2, \hat{h_0})^{z}_{\delta_1} \cdot e(g_1, \hat{h_0})^{z_s} \cdot \prod_{i=1}^{L} e(g_{i+1}, \hat{h_0})^{z_{m_i}}$$

**BBS+ 2006 Complexity Analysis**

- **Precomputed Values**
  - $*$ Base pairings: $(e(g_1, \hat{h_0}), e(g_2, \hat{h_0}), e(g_2, \hat{w}))$ 3 stored $\mathbb{G}_T$ elements
  - $*$ Message pairings: $\{e(g_{i+1}, \hat{h_0})\}_{i \in [1,L]}$ $L$ stored $\mathbb{G}_T$ elements
- **Prover's Initial Operations**
  - $*$ $(\delta_1, \delta_2) \leftarrow (r_1 e, r_2 e)$ $2F_p\times$
  - $*$ $(A_1, A_1^e, A_2) \leftarrow (g_1^{r_1} g_2^{r_2}, (g_1^{r_1} g_2^{r_2})^e, A g_2^{r_1})$ $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$
- **Prover Schnorr Operations**
  - $*$ Sample $L + 6$ field elements $(\tilde{r_1}, \tilde{r_2}, \tilde{e}, \tilde{\delta_1}, \tilde{\delta_2}, \tilde{s}, \tilde{m_1}, \ldots, \tilde{m_L})$ (negligible)
  - $*$ $(T_1, T_2) \leftarrow (g_1^{\tilde{r_1}} g_2^{\tilde{r_2}}, g_1^{\tilde{\delta_1}} g_2^{\tilde{\delta_2}})$ $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$
  - $*$ Compute $T_3$ using precomputed pairings $1P + (L + 4)E_{\mathbb{G}_T} + (L + 3)M_{\mathbb{G}_T}$
- **Proof Size / Data Sent**
  - $*$ $(A_1, A_1^e, A_2, T_1, T_2, T_3)$ 5 elements in $\mathbb{G}_1$, 1 element in $\mathbb{G}_T$
  - $*$ $z_x$ for $x \in \{r_1, r_2, e, \delta_1, \delta_2, s, m_1, \ldots, m_L\}$ $(L + 6)$ elements in $\mathbb{Z}_p^*$
- **Verifier Operations**
  - $*$ Commitment verification: $A_1^c \cdot T_1 \stackrel{?}{=} g_1^{z_{r_1}} g_2^{z_{r_2}}$ $3E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$
  - $*$ Exponentiation verification: $(A_1^e)^c \cdot T_2 \stackrel{?}{=} g_1^{z_{\delta_1}} g_2^{z_{\delta_2}}$ $3E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$
  - $*$ Pairing equation verification: $e(A_2, \widehat{w}) \stackrel{?}{=} e(g_2, \widehat{h_0}) \cdot T_3^c$ $2P + (L+4)E_{\mathbb{G}_T} + (L+3)M_{\mathbb{G}_T}$

### 16.11 BBS+ 2016 [CDL16]

asd

**Protocol for Signature Proof of Knowledge** asd

**Protocol 17: CDL16 Zero-Knowledge Proof of Knowledge of BBS+ [CDL16]**

**Relation $\mathcal{R}$**

$$\pi \leftarrow \mathsf{SPK}\{(m_1, \ldots, m_\ell), e, r_2, r_3, s') : \quad \frac{\bar{A}}{d} = A'^{-e} \cdot h_0^{r_2} \quad \wedge \quad g = d^{r_3} h_0^{-s'} \prod_{i=1}^{n} h_i^{-m_i}\}$$

**Common Input:** $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_2)$, $pk = (\hat{w}, \{h_i\}_{i=0}^{n})$

**Private Input:** $\mathcal{P}$ has BBS+ signature $(A, e, s)$ where $A = (gh_0^s \prod_{i=1}^{n} h_i^{m_i})^{\frac{1}{e+x}}$ and $b = gh_0^s \prod_1^n h_i^{m_i}$

1. $\mathcal{P}$ samples random $r_1, r_2 \leftarrow\$ \mathbb{Z}_p^*$, set $r_3 = \frac{1}{r_1}$, and $s' \leftarrow s - r_2 \cdot r_3$
   - Compute blinded signature: $A' \leftarrow A^{r_1}$ and set $\bar{A} \leftarrow A'^{-e} \cdot b^{r_1} \quad (= A'^x)$
   - compute $d \leftarrow (gh_0^s \prod_1^n h_i^{-m_i})^{r_1} \cdot h_0^{-r_2}$
   - sample Schnorr blinding factors from $\mathbb{Z}_p$
     $$\tilde{x} \text{ for } x \in \{r_1, r_2, e, s, m_1, \ldots, m_\ell\}$$
   - compute Schnorr Commitments:
     $$T_1 \leftarrow A^{\tilde{e}} \cdot h_0^{\tilde{s}} \qquad T_2 \leftarrow g \prod_{i=1}^{n} h_i^{\tilde{m}_i}$$
   - Send $(A', \bar{A}, d, T_1, T_2)$ to $\mathcal{V}$
2. $\mathcal{V}$ sends challenge $c \leftarrow\$ \mathbb{Z}_p^*$
3. $\mathcal{P}$ computes Schnorr responses:
   $$z_x = \tilde{x} + c \cdot x \text{ for } x \in \{r_1, r_2, e, s', m_1, \ldots, m_\ell\}$$
4. $\mathcal{V}$ checks
   $$A' \neq 1_{\mathbb{G}_1} \quad \wedge \quad e(A', \hat{w}) = e(\bar{A}, g_2) \qquad T_1 \overset{?}{=} A^{z_e} \cdot h_0^{z_s} \cdot (\bar{A})^{-c} \qquad T_2 \overset{?}{=} g^c \prod_{i=1}^{n} h_i^{z_{m_i}}$$

**Complexity Analysis**

- **Precomputed Values**
  - $*$ Base pairings: $(e(g, g_2), e(h_0, g_2))$ 2 stored $\mathbb{G}_T$ elements
  - $*$ Attribute pairings: $\{e(h_i, g_2)\}_{i\in[1,n]}$ $n$ stored $\mathbb{G}_T$ elements
- **Prover's Initial Operations**
  - $*$ Field computations: $(r_3, s') \leftarrow (\frac{1}{r_1}, s - r_2 \cdot r_3)$ $1F_p - +1F_p \times +1F_p+$
  - $*$ Signature randomization: $(A', \bar{A}) \leftarrow (A^{r_1}, A'^{-e} \cdot b^{r_1})$ $3E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1}$
  - $*$ Attribute commitment: $d \leftarrow (gh_0^{r_1} \prod_{i=1}^{n} h_i^{-m_i})^{r_2} \cdot h_0^{r_3}$ $(n+2)E_{\mathbb{G}_1} + (n+1)M_{\mathbb{G}_1}$
- **Prover Schnorr Operations**
  - $*$ Sample $n + 4$ field elements $(\tilde{e}, \tilde{s}, \tilde{m_1}, \ldots, \tilde{m_n}, \tilde{r_1}, \tilde{r_2})$ (negligible)
  - $*$ $(T_1, T_2) \leftarrow (A^{\tilde{e}} \cdot h_0^{\tilde{s}}, g \prod_{i=1}^{n} h_i^{\tilde{m_i}})$ $(n+2)E_{\mathbb{G}_1} + nM_{\mathbb{G}_1}$
- **Proof Size / Data Sent**
  - $*$ $(A', \bar{A}, d, T_1, T_2)$ 5 elements in $\mathbb{G}_1$
  - $*$ $z_x$ for $x \in \{r_1, r_2, e, s', m_1, \ldots, m_n\}$ $(n+4)$ elements in $\mathbb{Z}_p^*$
- **Verifier Operations**
  - $*$ Pairing verification: $e(A', \hat{w}) \stackrel{?}{=} e(\bar{A}, g_2)$ $2P + 1M_{\mathbb{G}_T}$
  - $*$ Commitment verification: $T_1 \stackrel{?}{=} A^{z_e} \cdot h_0^{z_s} \cdot (\bar{A})^{-c}$ $3E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$
  - $*$ Attribute verification: $T_2 \stackrel{?}{=} g^c \prod_{i=1}^{n} h_i^{z_{m_i}}$ $(n+1)E_{\mathbb{G}_1} + nM_{\mathbb{G}_1}$

### 16.12 CL04 [CL04]

Signature scheme D from [CL04] Mapped to Type3 Pairings

- **KeyGen**$(1^\lambda)$:
  - $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g}) \leftarrow\!\!\$\ \mathsf{BGGen}(1^\lambda)$
  - Sample $x, y \leftarrow\!\!\$\ \mathbb{Z}_p^*$, $(z_1, \ldots, z_\ell) \leftarrow\!\!\$\ (\mathbb{Z}_p^*)^\ell$. Set $sk = (x, y, z_1, \ldots, z_\ell)$
  - Compute public key:

$$\begin{aligned} X \leftarrow g^x \in \mathbb{G}_1, \quad \widehat{Y} \leftarrow \hat{g}^y \in \mathbb{G}_2 \qquad h \leftarrow\!\!\$\ \mathbb{Z}_p^*, H \leftarrow g^h \\ \widehat{Z}_i \leftarrow \hat{g}^{z_i} \in \mathbb{G}_2, \quad W_i \leftarrow g^{xz_i} \in \mathbb{G}_1 \quad \forall i \in [1..\ell] \end{aligned}$$

  - Output: $pk = (\mathsf{BG}, X, \widehat{Y}, \{\widehat{Z}_i\}_{i=1}^\ell, \{W_i\}_{i=1}^\ell)$
- **Sign**$(sk, (m_1, \ldots, m_\ell))$:
  - Sample $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$, compute $a \leftarrow g^\alpha \in \mathbb{G}_1$
  - For $i \in [1..\ell]$: $A_i \leftarrow a^{z_i} \in \mathbb{G}_1$
  - Compute $b \leftarrow a^y \in \mathbb{G}_1$, $B_i \leftarrow A_i^y \in \mathbb{G}_1$
  - Compute $c \leftarrow a^x \cdot b^\alpha \cdot \prod_{i=1}^\ell B_i^{m_i} \in \mathbb{G}_1$
  - Output $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$
- **Verify**$(pk, (m_1, \ldots, m_\ell), \sigma)$:
  - Parse $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$
  - Verify:

$$\begin{aligned} e(b, \hat{g}) &= e(a, \widehat{Y}) \\ \forall i : e(B_i, \hat{g}) &= e(A_i, \widehat{Y}) \\ e(c, \hat{g}) &= e(X, \hat{g}) \cdot e(a, \widehat{Y})^\alpha \cdot \prod_{i=1}^{\ell} e(A_i, \widehat{Y})^{m_i} \end{aligned}$$

- **BlindSign Protocol**:
  - **User** (with $(m_1, \ldots, m_\ell)$):
    - Sample $\gamma \leftarrow\!\!\$\ \mathbb{Z}_p^*$
    - Compute commitment: $\mathsf{cm} \leftarrow h^\gamma \prod_{i=1}^\ell W_i^{m_i} \in \mathbb{G}_1$
    - Send $\mathsf{cm}$ to Signer with proof:

$$\pi = \mathsf{PoK}\{(m_1, \ldots, m_\ell, \gamma) : \mathsf{cm} = H^\gamma \prod_{i=1}^{\ell} W_i^{m_i}\}$$

  - **Signer** (with $sk = (x, y, z_1, \ldots, z_\ell)$):
  - Verify $\pi$. If valid:

- Sample $\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$, compute $a \leftarrow g^{\alpha}$, $A_i \leftarrow a^{z_i}$
- Compute $b \leftarrow a^y$, $B_i \leftarrow A_i^y$
- Compute $c \leftarrow a^x \cdot \mathsf{cm}^{xy} \in \mathbb{G}_1$
- Send $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ to User

∗ **User** (Unblinding):

- Sample $r, r' \leftarrow\!\!\$\ \mathbb{Z}_p^*$
- Compute blinded signature $\tilde{\sigma}$:

$$\begin{aligned} \tilde{a} &\leftarrow a^{r'} \\ \tilde{A}_i &\leftarrow A_i^{r'} && \text{for } i \in [1, \ell] \\ \tilde{b} &\leftarrow b^{r'} \\ \tilde{B}_i &\leftarrow B_i^{r'} && \text{for } i \in [1, \ell] \\ \tilde{c}^r &\leftarrow c^{rr'} \end{aligned}$$

∗ **ZKPoK-Verify**:

- **Common Input**: $pk$, $\tilde{\sigma} = (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \tilde{c})$
- **Prover** (with $m_1, \ldots, m_\ell$):
- Compute pairings:

$$\begin{aligned} v_x &\leftarrow e(\tilde{a}, \hat{g}) \\ v_{xy} &\leftarrow e(\tilde{a}, \widehat{Y}) \\ v_s &\leftarrow e(\tilde{c}, \hat{g}) \end{aligned}$$

- Generate proof:

$$\pi = \mathsf{PoK}\{(m_1, \ldots, m_\ell, r) : v_s = v_x \cdot \prod_{i=1}^{\ell} e(\tilde{A}_i, \widehat{Y})^{m_i}\}$$

- **Verifier**:
- Check signature structure:

$$\begin{aligned} e(\tilde{b}, \hat{g}) &\stackrel{?}{=} e(\tilde{a}, \widehat{Y}) \\ e(\tilde{B}_i, \hat{g}) &\stackrel{?}{=} e(\tilde{A}_i, \widehat{Y}) && \text{for } i \in [1, \ell] \end{aligned}$$

- Verify proof $\pi$ using the pairing equation for $v_s$
- Accept if all checks pass

- **Precomputed Values**
  - Base pairings: $(e(\tilde{a}, \hat{g}), e(\tilde{a}, \widehat{Y}))$ 2 stored $\mathbb{G}_T$ elements
- **Prover's Initial Operations (Signature Randomization)**
  - $(\tilde{a}, \tilde{b}, \{\tilde{A}_i\}, \{\tilde{B}_i\}, \tilde{c}) \leftarrow (a^{r'}, b^{r'}, \{A_i^{r'}\}, \{B_i^{r'}\}, c^{rr'})$ $(2\ell + 3)E_{\mathbb{G}_1}$
- **Prover Schnorr Operations**
  - Sample $\ell + 1$ field elements $(\tilde{m_1}, \ldots, \tilde{m_\ell}, \tilde{r})$ (negligible)
  - Compute commitment $T \leftarrow v_x \cdot \prod_{i=1}^{\ell} e(\tilde{A}_i, \widehat{Y})^{\tilde{m_i}}$ $\ell E_{\mathbb{G}_T} + (\ell - 1)M_{\mathbb{G}_T}$
- **Proof Size / Data Sent**
  - $(\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \tilde{c}, T)$ $(2\ell + 3)$ elements in $\mathbb{G}_1$, 1 element in $\mathbb{G}_T$
  - $z_x$ for $x \in \{m_1, \ldots, m_\ell, r\}$ $(\ell + 1)$ elements in $\mathbb{Z}_p^*$
- **Verifier Operations**
  - Structure verification: $e(\tilde{b}, \hat{g}) \stackrel{?}{=} e(\tilde{a}, \widehat{Y})$ $2P$
  - Attribute consistency: $\{e(\tilde{B}_i, \hat{g}) \stackrel{?}{=} e(\tilde{A}_i, \widehat{Y})\}_{i=1}^{\ell}$ $2\ell P$
  - Challenge verification: $v_s \stackrel{?}{=} v_x \cdot \prod_{i=1}^{\ell} e(\tilde{A}_i, \widehat{Y})^{z_{m_i}}$ $\ell P + \ell E_{\mathbb{G}_T} + (\ell - 1)M_{\mathbb{G}_T}$

## 16.13 Analysis No Optimization

| Protocol | Show (Prover) | Verify (Verifier) | Data Sent |
|---|---|---|---|
| CL04 | **Randomize:**<br>• $(2\ell+3)E_{\mathbb{G}_1}$ (sig. rand.)<br>**Schnorr Proof:**<br>• $(\ell+1)$ samplings<br>• $\ell E_{\mathbb{G}_T} + (\ell-1)M_{\mathbb{G}_T}$ (commits) | • $(2\ell+2)P$ (struct. verify)<br>• $\ell P + \ell E_{\mathbb{G}_T} + (\ell-1)M_{\mathbb{G}_T}$ (chall. verify) | • $(2\ell+3)\mathbb{G}_1$<br>• $1\mathbb{G}_T$<br>• $(\ell+1)\mathbb{Z}_p^*$ |
| BBS+ | **Randomize:**<br>• $2F_p\times$ (deltas)<br>• $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$ (sig. rand.)<br>**Schnorr Proof:**<br>• $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$ (commits)<br>• $1P + (L+4)E_{\mathbb{G}_T} + (L+3)M_{\mathbb{G}_T}$ | • $6E_{\mathbb{G}_1} + 4M_{\mathbb{G}_1}$ (commits)<br>• $2P + (L+4)E_{\mathbb{G}_T} + (L+3)M_{\mathbb{G}_T}$ | • $5\mathbb{G}_1$<br>• $1\mathbb{G}_T$<br>• $(L+6)\mathbb{Z}_p^*$ |
| BBS+ 2016 | **Randomize:**<br>• $1F_p^- + 2F_p\times$ (field ops)<br>• $3E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1}$ (sig. rand.)<br>• $(n+2)E_{\mathbb{G}_1} + (n+1)M_{\mathbb{G}_1}$ (attr. commit)<br>**Schnorr Proof:**<br>• $(n+4)$ samplings<br>• $(n+2)E_{\mathbb{G}_1} + nM_{\mathbb{G}_1}$ (commits) | • $2P + 1M_{\mathbb{G}_T}$ (pairing)<br>• $(n+4)E_{\mathbb{G}_1} + (n+2)M_{\mathbb{G}_1}$ (commits) | • $5\mathbb{G}_1$<br>• $(n+4)\mathbb{Z}_p^*$ |
| PS16 | **Randomize:**<br>• $3E_{\mathbb{G}_1} + 1M_{\mathbb{G}_1}$ (sig. rand.)<br>**Schnorr Proof:**<br>• $(\ell+2)P + (\ell+1)E_{\mathbb{G}_T} + (\ell+1)M_{\mathbb{G}_T}$ | • $(\ell+3)P$<br>• $(\ell+1)E_{\mathbb{G}_T}$<br>• $(\ell+2)M_{\mathbb{G}_T}$ | • $2\mathbb{G}_1$<br>• $1\mathbb{G}_T$<br>• $(\ell+2)\mathbb{Z}_p^*$ |
| PS UTT | **Randomize:**<br>• $4E_{\mathbb{G}_1} + 2M_{\mathbb{G}_1}$<br>• $1E_{\mathbb{G}_2} + 1M_{\mathbb{G}_2}$<br>**Schnorr Proof:**<br>• $(\ell+1)E_{\mathbb{G}_1} + \ell M_{\mathbb{G}_1}$<br>• $1E_{\mathbb{G}_1} + 1M_{\mathbb{G}_2} + 1P$ | • $(\ell+3)E_{\mathbb{G}_1} + (\ell+2)M_{\mathbb{G}_1} + 1M_{\mathbb{G}_2}$<br>• $1E_{\mathbb{G}_T} + 2M_{\mathbb{G}_T}$<br>• $4P$ | • $5\mathbb{G}_1$<br>• $1\mathbb{G}_2$<br>• $(\ell+2)\mathbb{Z}_p$ |

### 16.14 Optimizations

Most cryptographic papers analyze computational complexity in a theoretical sense without regarding optimizations available for operations within their credentials. For example, some schemes measure the complexity of multiple pairing operations [PS16] but do not take into account optimizations such as computing Miller Loop and Final Exponentiation separately, algorithms available in practical cryptography libraries. We discuss optimizations found in many cryptography libraries below and the impact they have in practical implementations.

- **Pairing Optimizations:**
  - ∗ **Miller Loop + Final Exponentiation:** A pairing computation (e.g., Tate, Ate) consists of two distinct phases, denoted as $e(P, Q)$ in complexity analysis:
    - · *Miller Loop*: Computes a rational function over points (complexity linear in loop length)
    - · *Final Exponentiation*: Ensures unique representation in the target group $\mathbb{G}_T$

    For products of pairings, we can optimize:

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

    The left-hand side requires:
    - · $\ell$ complete pairings ($\ell P$)
    - · $(\ell - 1)$ multiplications in $\mathbb{G}_T$ ($(\ell - 1)M_{\mathbb{G}_T}$)

    The right-hand side reduces to:
    - · $\ell$ Miller Loops ($\approx 0.4\ell P$)
    - · 1 Single Final Exponentiation ($\approx 0.6P$)
  - ∗ **Batch Pairing Inversion:** For equations of the form $e(a, b) = e(c, d)$, transform to:

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

    This optimization reduces standalone pairing counts by approximately 50% through the elimination of redundant Final Exponentiations.
- **Multi-Scalar Multiplication (MSM):** Multi-scalar multiplication addresses the common operation of computing:

  $$T = \prod_{i=1}^{\ell} g_i^{m_i}$$

  where $g_i$ are group elements and $m_i$ are scalar exponents. A naive approach computing each $g_i^{m_i}$ separately would require $\ell$ independent exponentiations. However, several algorithms exist that can compute this product significantly faster by processing multiple scalars simultaneously, similar to how carrying works in manual addition. These optimizations reduce the complexity from $O(\ell)$ exponentiations to sublinear complexity in $\ell$.

  This is useful in Anonymous Credentials, we use it with:
  - ∗ Commitment openings involving multiple attributes
  - ∗ Proof generation with multiple blinding factors

∗ Verification equations combining multiple witnesses

All schemes we denote benefit from MSM.

- **Precomputation Strategies:**

  ∗ Cache frequently used pairing results: $e(g_i, \hat{h}_j)$ for fixed bases

  ∗ Optimize BBS+ operations: Transform $\prod_{i=1}^{L} e(g_{i+1}, \hat{h}_0)^{\tilde{m}_i}$ into $L$ multiplications in $\mathbb{G}_T$

  ∗ Store processed group elements for repeated operations

- **Batch Verification:**

  ∗ For $N$ BBS+ proofs under a common public key:

$$\prod_{i=1}^{N} e(A_i, \hat{w}_i) \stackrel{?}{=} e(g, \hat{h}_0)^N \cdot \prod_{j=1}^{n} e(g_j, \hat{h}_j)^{\sum_{i=1}^{N} m_{i,j}}$$

  This optimization reduces complexity from $O(N \cdot n)$ to $O(n + N)$ pairings

**Practical Implications**

- **PS2016:** Leverages pairing batching optimizations effectively but requires q-type assumptions
- **BBS+:** Employs precomputation and MSM optimizations for attribute handling, though proof size remains linear
- **PSUTT:** Achieves balance between succinctness and token updatability through careful application of MSM optimizations

## 17 Improvements

**$\mathbb{G}_1$ Signature** We change the signature algorithm from signing in G1 to signing in G2, this gives a bigger signature (2 x G2 instead of 2 x G1) but faster verification.

$\mathsf{RS.Ver}(\sigma, \widetilde{\mathsf{cm}'}, ..)$ takes in $\sigma_2 \in \mathbb{G}_1$, and $\widetilde{\mathsf{cm}'} \in \mathbb{G}_2$. My intuition is we need to do the pairing equality check $e(\mathsf{cm}', \tilde{g}) = e(g, \widetilde{\mathsf{cm}'})$ to be able to use $\mathsf{cm} \in \mathbb{G}_1$ for $\mathsf{PoK}$ because it's not explicitly used in the pairing verification, the $\mathbb{G}_2$ commitment is. The verifier needs to know why it can trust it.

- $\mathsf{CM.Setup}$ :$\mathsf{ck} \leftarrow (g, (g_1, \dots, g_\ell), \tilde{g}, (\tilde{g}_1, \dots, \tilde{g}_\ell))$
- $\mathsf{RS.KeyGen}$ :$(\mathsf{sk}, \widetilde{\mathsf{vk}}) \leftarrow (g^x, \tilde{g}^x)$, return $(\mathsf{sk}, \widetilde{\mathsf{vk}})$
- $\mathsf{RS.Sign}$ : $\sigma \in \mathbb{G}_1 \leftarrow (\sigma_1, \sigma_2) = (h, (\mathsf{sk} \cdot \mathsf{cm})^u)$
- $\mathsf{RS.Rerand}$ : $\quad {\sigma_1}' \leftarrow {\sigma_1}^{u_\Delta} \quad {\sigma_2}' \leftarrow (\sigma_2 \cdot {\sigma_1}^{r_\Delta})^{u_\Delta} \quad \mathsf{cm}' \leftarrow \mathsf{cm} \cdot g^{r_\Delta} \quad \widetilde{\mathsf{cm}}' \leftarrow \widetilde{\mathsf{cm}} \cdot \tilde{g}^{r_\Delta}$
- $\mathsf{RS.Ver}(\widetilde{\mathsf{vk}}, \mathsf{cm}', \sigma') \rightarrow \{0, 1\}$ :

$$e(\sigma_2', \tilde{g}) = e(h, \mathsf{vk} \cdot \widetilde{\mathsf{cm}'}) \qquad \wedge \qquad e(\mathsf{cm}', \tilde{g}) = e(g, \widetilde{\mathsf{cm}'})$$

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

**$\mathbb{G}_2$ Signature** When $\sigma_2 \in \mathbb{G}_2$, and $\mathsf{RS.Ver}(\mathsf{cm}' \in \mathbb{G}_1, \tilde{\sigma} \in \mathbb{G}_2)$ is a pairing verification with $\mathsf{cm}' \in \mathbb{G}_1$. Using $\mathsf{cm}' \in \mathbb{G}_1$ we can follow with $\mathsf{PoK}$

- $\mathsf{CM.Setup}$ :$\mathsf{ck} \leftarrow (g, (g_1, \dots, g_\ell), \tilde{g}, (\tilde{g}_1, \dots, \tilde{g}_\ell))$
- $\mathsf{RS.KeyGen}$ :$(\widetilde{\mathsf{sk}}, \mathsf{vk}) \leftarrow (\tilde{g}^x, g^x)$, return $(\widetilde{\mathsf{sk}}, \mathsf{vk}))$
- $\mathsf{RS.Sign}$ : $\tilde{\sigma} \in \mathbb{G}_2 \leftarrow (\widetilde{\sigma_1}, \widetilde{\sigma_2}) = (\tilde{h}, (\widetilde{\mathsf{sk}} \cdot \widetilde{\mathsf{cm}})^u)$
- $\mathsf{RS.Rerand}$ : $\quad \widetilde{\sigma_1}' \leftarrow \widetilde{\sigma_1}^{u_\Delta} \quad \widetilde{\sigma_2}' \leftarrow (\widetilde{\sigma_2} \cdot \widetilde{\sigma_1}^{r_\Delta})^{u_\Delta} \quad \mathsf{cm}' \leftarrow \mathsf{cm} \cdot g^{r_\Delta} \quad \widetilde{\mathsf{cm}}' \leftarrow \widetilde{\mathsf{cm}} \cdot \tilde{g}^{r_\Delta}$
- $\mathsf{RS.Ver}(\mathsf{vk}, \mathsf{cm}', \tilde{\sigma}') \rightarrow \{0, 1\}$ :

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

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

**$\mathbb{G}_2$ Correctness**

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

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

2. then PoK

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

| Scheme | Show | | Verify | | |
|---|---|---|---|---|---|
| | Credential | PoK | Credential | PoK | Data Sent |
| PSUTT | $4E_{\mathbb{G}_1} + 1E_{\mathbb{G}_2}$ | $1E_{\mathbb{G}_1} + 1P$ $MSM(\ell+1)\mathbb{G}_1$ | $1E_{\mathbb{G}_T}$ $4M\text{Loop}\mathbb{G}_T + 1FE_{\mathbb{G}_T}$ | $MSM(\ell+2)_{\mathbb{G}_1}$ | $5\mathbb{G}_1 + 1\mathbb{G}_2 + (\ell+2)\mathbb{Z}_p$ |
| PSUTT+LVS | $3E_{\mathbb{G}_2} + 1E_{G_1}$ | $1E_{G_1}+$ $MSM(\ell+1)_{\mathbb{G}_1}$ | $2M\text{Loop}\mathbb{G}_T + 1FE_{\mathbb{G}_T}$ | $MSM(\ell+1)_{\mathbb{G}_1} + 1E_{\mathbb{G}_1}$ | $2\mathbb{G}_1 + 2\mathbb{G}_2 + (\ell+1)\mathbb{Z}_p$ |

Table 5: Removed scalar point multiplications and anything in $\mathbb{F}_p$

**Optimizations moving to $\sigma \in \mathbb{G}_2$**

- Show Cred changes from 4EG1 + 1EG2 to 3EG2 + 1EG1 ()
- PoK changes from $1E_{\mathbb{G}_1} + 1P + MSM(\ell+1)\mathbb{G}_1 \quad \Rightarrow \quad 1E_{G_1} + MSM(\ell+1)_{\mathbb{G}_1}$ — We remove the pairing!
- Verify cred changes from $1E_{\mathbb{G}_T} + 4M\text{Loop}\mathbb{G}_T + 1FinalExp_{\mathbb{G}_T} \quad \Rightarrow \quad 2M\text{Loop}\mathbb{G}_T + 1FinalExp_{\mathbb{G}_T}$ — reduce 2 miller loop and 1 GT exponentiation

## 18 Old Reductions

### 18.1 Unforgeability

- Challenger Setup
- Adversary Query Oracle
- Simulation / Forgery
- Win Condition

The game tracks $\mathsf{CRED}, \mathsf{COM}, \mathsf{OWNR}$, $\mathcal{A}$ has access to oracles $(\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Show}})$ modelling the adversaries real-world actions.

Lists $\mathsf{CRED}, \mathsf{OWNR}, \mathsf{COM}$ let us define a forgery, by EUF-CMA definition, if $verify(\mathsf{cred_m}, \mathsf{cm_m}, \mathsf{opk_m}) = 1$ and $\mathsf{cred_m} \times \mathsf{cm_m} \notin \mathsf{CRED} \times \mathsf{COM}$ then it's a forgery!. Also used for reductions to primitives.

The Unforgeability Adversary has access to

- $\mathcal{O}_{\mathsf{HU}}$ : $\mathcal{A}$ creates honest users => Create honest users that $\mathcal{A}$ can interact with but not control
- $\mathcal{O}_{\mathsf{CU}}$ : allows an adversary to corrupt a user, like hacking a user account and revealing their secrets => models insider threats, users whose secrets are known to $\mathcal{A}$
- $\mathcal{O}_{\mathsf{ObtIss}}$ : models $\mathcal{A}$ observing legitimate credential issuance for message $m$. => $\mathcal{A}$ controls inputs, execution must follow protocol
- $\mathcal{O}_{\mathsf{Show}}$ : shows legitimate credential usage satisfying a predicate $\phi$ => $\mathcal{A}$ learns what's revealed in showing protocol

### 18.2 Basic Unforgeability Version 1

Win Condition: Adversary outputs $\mathsf{cred}^*, m^*$ where $\mathsf{Verify}(\mathsf{cred}^*, m^*, \mathsf{opk}) = 1$ and $m^* \notin \mathsf{MSG}$ the signed messages

```
$\mathrm{Exp}^{\mathsf{EUF\text{-}CMA}}(\mathcal{A})$
    // Challenger Setup
Initialize $\mathsf{HU}, \mathsf{CRED}, \mathsf{MSG} = \emptyset$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
    // $\mathcal{A}$ queries oracles
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{ObtIss}}}(\mathsf{opk})$
$\mathcal{A}$ outputs forgery $(\mathsf{cred}^*, m^*)$
    // $\mathcal{A}$ wins if
$\mathsf{Verify}(\mathsf{cred}^*, m^*, \mathsf{opk}) = 1 \wedge$
$m^* \notin \mathsf{MSG} \ \wedge \ \mathsf{cred}^* \notin \mathsf{CRED}[i]$ for all $i$

$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$
return Honest User $i$

$\mathcal{O}_{\mathsf{ObtIss}}(i, m)$
if $i \notin \mathsf{HU}, \perp$
$\mathsf{MSG}[i] = m$
$\mathsf{CRED}[i] = (\mathsf{cred}, m)$
$\mathsf{cred} = (\mathsf{Obtain}(\mathsf{usk}[i], m), \mathsf{Issue}(\mathsf{osk}, m))$
return $\mathsf{cred}$
```

Fig. 8: The EUF-CMA game, single credential, step 1

### 18.3 Basic Unforgeability Version 2

We change to sets because we use a multi-credential multi-issuer environment. HU is a set of honest user indices $\{i_1, \ldots\}$

Win Condition: Adversary outputs $\mathsf{cred}^*, m^*$ where $\mathsf{Verify}(\mathsf{cred}^*, m^*, \mathsf{opk}) = 1$ and $m^* \notin \mathsf{MSG}$ the signed messages

```
$\mathrm{Exp}^{\mathsf{EUF\text{-}CMA}}(\mathcal{A})$
    // Challenger Setup
Initialize $\mathsf{HU}, \mathsf{CRED}, \mathsf{MSG} = \emptyset$
 where $\mathsf{CRED} \subseteq \{(\mathsf{cred}, m)\}, \mathsf{MSG} \subseteq \{m\}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
    // $\mathcal{A}$ queries oracles
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{ObtIss}}}(\mathsf{opk})$
$\mathcal{A}$ outputs forgery $(\mathsf{cred}^*, m^*)$
    // $\mathcal{A}$ wins if
$\mathsf{Verify}(\mathsf{cred}^*, m^*, \mathsf{opk}) = 1 \wedge$
$m^* \notin \mathsf{MSG} \ \wedge \ \mathsf{cred}^* \notin \mathsf{CRED}$

$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$
return Honest User $i$

$\mathcal{O}_{\mathsf{ObtIss}}(i, m)$
if $i \notin \mathsf{HU}, \perp$
$\mathsf{cred} = (\mathsf{Obtain}(\mathsf{usk}[i], m), \mathsf{Issue}(\mathsf{osk}, m))$
$\mathsf{CRED} = \mathsf{CRED} \cup \{(\mathsf{cred}, m)\}$
$\mathsf{MSG} = \mathsf{MSG} \cup \{m\}$
return $\mathsf{cred}$
```

Fig. 9: The EUF-CMA game, single credential, step 2

### 18.4 Basic Unforgeability Version 3 - Adding Commitments and ZKP

Win Condition: Adversary outputs $\mathsf{cred}^*, m^*$ where $\mathsf{Verify}(\mathsf{cred}^*, m^*, \mathsf{opk}) = 1$ and $m^* \notin \mathsf{MSG}$ the signed messages

```
$\mathrm{Exp}^{\mathsf{UNF\text{-}1}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda)$
    // Challenger Setup
$\text{Initialize } \mathsf{HU}, \mathsf{CRED}, \mathsf{MSG} = \emptyset$
$\mathsf{CRED} \subseteq \{(\mathsf{cred}, m)\}, \mathsf{MSG} \subseteq \{m\}$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \mathsf{ck} \leftarrow \mathsf{CM.Com}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
    // $\mathcal{A}$ queries oracles
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{Obtiss}}}(\mathsf{opk})$
$\mathcal{A} \text{ outputs forgery } (\mathsf{cred}^*, \mathsf{cm}^*, \mathbf{m}^*, \mathsf{usk}^*, \pi^*)$
    // $\mathcal{A}$ wins if
$\Pi^{\mathcal{R}_{\mathsf{sok}}}(\mathsf{cred}^*, \mathsf{cm}^*, \mathbf{m}^*, \mathsf{usk}^*) = 1 \text{ such that}$
$\mathsf{cm}^* = \mathsf{CM.Com}([m^*]; \mathsf{usk}^*) \wedge$
$\mathsf{RS.Ver}(\mathsf{cred}^*, \mathsf{cm}^*, \mathsf{opk}) = 1 \wedge$
$(\mathsf{cred}^*, \mathsf{cm}^*, \mathbf{m}^*) \notin \mathsf{CRED}$

$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$
$\mathbf{return} \text{ Honest User } i$

$\mathcal{O}_{\mathsf{Obtiss}}(i, m)$
$\mathbf{if}\ i \notin \mathsf{HU}, \bot$
$\mathsf{usk} \in \mathbb{Z}_p \leftarrow\!\!\$\ \mathsf{UserKeyGen}(1^\lambda)$
$\mathsf{cm} \leftarrow \mathsf{CM.Com}([m]; \mathsf{usk})$
$\Pi^{\mathcal{R}_{\mathsf{s.disclose}}} = \mathsf{ZKPoK}\{(m, \mathsf{usk}) | \mathsf{cm} = \mathsf{CM.Com}([m]; \mathsf{usk})\}$
$\mathsf{cred} = (\mathsf{Obtain}(\mathsf{usk}[i], m), \mathsf{Issue}(\mathsf{osk}, m))$
$\mathbf{if}\ \mathsf{ZK.Verify}(\Pi^{\mathcal{R}_{\mathsf{s.disclose}}}, \mathsf{cm}) = 0, return \bot$
$\mathsf{CRED} := \mathsf{CRED} \cup \{(\mathsf{cred}, \mathsf{cm}, m)\}$
$\mathsf{MSG} = \mathsf{MSG} \cup \{m\}$
$\mathbf{return}\ (\mathsf{cred}, \mathsf{cm})$
```

Fig. 10: The UNF-1 game, single credential, step 3

### 18.5 Type 1 Unforgeability V4

Changes

- Changing $m$ to $aux$ to include proofs later
- adding Corrupt User list CU
- changing game to UNF-1

```
$\mathrm{Exp}^{\mathsf{UNF\text{-}1}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda)$
    // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\text{For each issuer } i \in [n] : (\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
$\text{Initialize } \mathsf{HU}, \mathsf{CUCRED}, \mathsf{OWNR} = \emptyset$
$\mathcal{A} \text{ has oracle access } \mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{Obtiss}}}(\{\mathsf{opk}_i\}_{i \in [n]})$
    // Winning Condition
$\mathcal{A} \text{ outputs } (\mathsf{cred}^*, j) \text{ where :}$
    $\exists\ i \text{ such that } \mathsf{Verify}(\mathsf{cred}^*, \mathsf{opk}_i) = 1 \text{ AND}$
    Either
        $\mathsf{cred}^* \notin \mathsf{CRED}$ // Fresh Forgery
     OR
        $\mathsf{OWNR}[\mathsf{cred}^*] \in \mathsf{HU} \setminus \mathsf{CU}$ // Stolen Credential

$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i, \mathbf{return}\ i$

$\mathcal{O}_{\mathsf{CU}}(i)$
    // Can only corrupt Honest User
$\mathbf{if}\ i \in \mathsf{HU}, \mathsf{CU}[i] = i$
$\mathbf{return}\ \text{ all } \mathsf{cred}_i \text{ owned by } i$

$\mathcal{O}_{\mathsf{Obtiss}}(i, aux)$
$\mathbf{if}\ i \notin \mathsf{HU} \vee i \in \mathsf{CU}, \bot$
$\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], aux), \mathsf{Issue}(\mathsf{osk}, aux))$
$\text{Add } (\mathsf{cred}, i) to (\mathsf{CRED}, \mathsf{OWNR})$
$\mathbf{return}\ \mathsf{cred}$
```

Fig. 11: The

### 18.6 Type 1 Unforgeability V5

Changes

- Adding ISSUER issuer state tracking for multi credential multi issuer environment
- Adding PARENT mapping context credentials to master credentials

```
Exp^{UNF-1}_{MIMC-ABC,A}(λ)
  // Challenger Setup
pp ← Setup(1^λ)
For each issuer $i \in [n]$ : $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
Initialize $\mathsf{HU}, \mathsf{CU}, \mathsf{CRED}, \mathsf{OWNR} = \emptyset$
$\mathcal{A}$ has oracle access $\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}}(\{\mathsf{opk}_i\}_{i \in [n]})$
  // Winning Condition
$\mathcal{A}$ outputs $(\mathsf{cred}^*, j)$ where :
  $\exists\, i$ such that $\mathsf{Verify}(\mathsf{cred}^*, \mathsf{opk}_i) = 1$ AND
    $\mathsf{cred}^* \notin \mathsf{CRED}$    // Fresh Forgery
  OR
    $\mathsf{OWNR}[\mathsf{cred}^*] \in \mathsf{HU} \setminus \mathsf{CU}$    // Stolen Credential
  OR
    $\mathsf{ISSUER}[\mathsf{cred}^*] \neq j$    // Wrong Issuer

$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$, return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
  // Can only corrupt Honest User
if $i \in \mathsf{HU}$, $\mathsf{CU}[i] = i$
return  all $\mathsf{cred}_i$ owned by $i$

$\mathcal{O}_{\mathsf{ObtIss}}(i, aux)$
if $i \notin \mathsf{HU} \lor i \in \mathsf{CU}, \perp$
$\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], aux), \mathsf{Issue}(\mathsf{osk}, aux))$
Add $(\mathsf{cred}, i) to (\mathsf{CRED}, \mathsf{OWNR})$
return cred
```

Fig. 12: The

## 18.7 Type 1 Unforgeability V6

Changes

- Adding CTX mapping credentials to their context
- adding connection between master and context credential in ObtIss

```
$\mathrm{Exp}^{\mathsf{UNF\text{-}1}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda)$

  // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
For each issuer $i \in [n]$ : $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
Initialize $\mathsf{HU}, \mathsf{CU}, \mathsf{CRED}, \mathsf{OWNR}, \mathsf{ISSUER}, \mathsf{CTX} = \emptyset$
$\mathcal{A}$ has oracle access $\mathcal{A}^{\mathcal{O}_{\mathsf{HU}},\mathcal{O}_{\mathsf{CU}},\mathcal{O}_{\mathsf{ObtIss}}}(\{\mathsf{opk}_i\}_{i\in[n]})$
  // Winning Condition
$\mathcal{A}$ outputs $(\mathsf{cred}^*, i^*, j)$ where :
  $\mathsf{Verify}(\mathsf{cred}^*, \mathsf{opk}_j, \mathsf{ctx}^*) = 1$ AND
    $\mathsf{cred}^* \notin \mathsf{CRED}$   // Fresh Forgery
   OR
    $\mathsf{OWNR}[\mathsf{cred}^*] \in \mathsf{HU} \setminus \mathsf{CU}$   // Stolen Credential
   OR
    $\mathsf{ISSUER}[\mathsf{cred}^*] \neq j$   // Wrong Issuer
   OR
    $\mathsf{CTX}[\mathsf{cred}^*] \neq \mathsf{ctx}^*$   // Wrong Context
```

```
$\mathcal{O}_{\mathsf{HU}}(i)$

$\mathsf{HU}[i] = i$, return $i$
```

```
$\mathcal{O}_{\mathsf{CU}}(i)$

  // Can only corrupt Honest User
if $i \in \mathsf{HU}, \mathsf{CU}[i] = i$
return  all $\mathsf{cred}_i$ owned by $i$
```

```
$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathsf{ctx}, aux)$

if $i \notin \mathsf{HU} \vee i \in \mathsf{CU}, \perp$
if $\mathsf{ctx} \neq$ "$master$" :
  Require $\exists\ \mathsf{cred}_\mathsf{m}$ where
  $\mathsf{CTX}[\mathsf{cred}_\mathsf{m}] =$ "$master$" AND $\mathsf{OWNR}[\mathsf{cred}_\mathsf{m}] = i$
$\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], \mathsf{ctx}, aux), \mathsf{Issue}(\mathsf{osk}, \mathsf{ctx}, aux))$
Add $(\mathsf{cred}, i, \mathsf{ctx}) to (\mathsf{CRED}, \mathsf{OWNR}, \mathsf{CTX})$
return $\mathsf{cred}$
```

Fig. 13: V6

## 18.8 Type 1 Unforgeability V7

Changes

- Adding CTX mapping credentials to their context
- adding connection between master and context credential in ObtIss

```
$\mathrm{Exp}^{\mathsf{UNF\text{-}1}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda)$
  // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
For each issuer $i \in [n]$ : $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
Initialize $\mathsf{HU}, \mathsf{CU}, \mathsf{CRED}, \mathsf{OWNR}, \mathsf{ISSUER}, \mathsf{CTX}, \mathsf{PARENT} = \emptyset$
$\mathcal{A}$ has oracle access $\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}}(\{\mathsf{opk}_i\}_{i \in [n]})$
  // Winning Condition
$\mathcal{A}$ outputs $(\mathsf{cred}^*, i^*, j, \mathsf{ctx})$ where :
  $\mathsf{Verify}(\mathsf{cred}^*, \mathsf{opk}_j, \mathsf{ctx}^*) = 1$ AND
    $\mathsf{cred}^* \notin \mathsf{CRED}$   // Fresh Forgery
   OR
    $\mathsf{OWNR}[\mathsf{cred}^*] \in \mathsf{HU} \setminus \mathsf{CU}$   // Stolen Credential
   OR
    $\mathsf{ISSUER}[\mathsf{cred}^*] \neq j$   // Wrong Issuer
   OR
    $\mathsf{CTX}[\mathsf{cred}^*] \neq \mathsf{ctx}^*$   // Wrong Context
```

```
$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$, return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
  // Can only corrupt Honest User
if $i \in \mathsf{HU}, \mathsf{CU}[i] = i$
return  all $\mathsf{cred}_i$ owned by $i$

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathsf{ctx}, aux)$
if $i \notin \mathsf{HU} \vee i \in \mathsf{CU}, \perp$
if $\mathsf{ctx} \neq$ "$master$" :
  Require $\exists\ \mathsf{cred}_{\mathsf{m}}$ where
  $\mathsf{CTX}[\mathsf{cred}_{\mathsf{m}}] =$ "$master$" AND $\mathsf{OWNR}[\mathsf{cred}_{\mathsf{m}}] = i$
$\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], \mathsf{ctx}, aux), \mathsf{Issue}(\mathsf{osk}, \mathsf{ctx}, aux))$
Add $(\mathsf{cred}, i, \mathsf{ctx}) to (\mathsf{CRED}, \mathsf{OWNR}, \mathsf{CTX})$
return $\mathsf{cred}$
```

Fig. 14: V7

## 18.9 Type 1 Unforgeability V8

Changes

- Adding ID mapping credentials to user identifier
- adding connection between master and context credential in ObtIss

```
$\mathrm{Exp}^{\mathsf{UNF\text{-}1}}_{\mathsf{MIMC\text{-}ABC},\mathcal{A}}(\lambda)$
  // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
For each issuer $i \in [n]$ : $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
Initialize $\mathsf{HU}, \mathsf{CU}, \mathsf{CRED}, \mathsf{OWNR}, \mathsf{ISSUER}, \mathsf{CTX}, \mathsf{PARENT} = \emptyset$
$\mathcal{A}$ has oracle access $\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}}(\{\mathsf{opk}_i\}_{i \in [n]})$
  // Winning Condition
$\mathcal{A}$ outputs $(\mathsf{cred}^*, i^*, j, \mathsf{ctx})$ where :
  $\mathsf{Verify}(\mathsf{cred}^*, \mathsf{opk}_j, \mathsf{ctx}^*) = 1$ AND
    $\mathsf{cred}^* \notin \mathsf{CRED}$   // Fresh Forgery
   OR
    $\mathsf{OWNR}[\mathsf{cred}^*] \in \mathsf{HU} \setminus \mathsf{CU}$   // Stolen Credential
   OR
    $\mathsf{ISSUER}[\mathsf{cred}^*] \neq j$   // Wrong Issuer
   OR
    $\mathsf{CTX}[\mathsf{cred}^*] \neq \mathsf{ctx}^*$   // Wrong Context
```

```
$\mathcal{O}_{\mathsf{HU}}(i)$
$\mathsf{HU}[i] = i$, return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
  // Can only corrupt Honest User
if $i \in \mathsf{HU}, \mathsf{CU}[i] = i$
return all $\mathsf{cred}_i$ owned by $i$

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathsf{ctx}, aux)$
if $i \notin \mathsf{HU} \vee i \in \mathsf{CU}, \perp$
if $\mathsf{ctx} = "master"$ :
  Require $\exists\ \mathsf{cred}_\mathsf{m}$ where
  $\mathsf{PARENT}[\mathsf{cred}_\mathsf{m}] = \perp$ AND
  $\mathsf{OWNR}[\mathsf{cred}_\mathsf{m}] = i$ AND
  $\mathsf{ID}[\mathsf{cred}_\mathsf{m}] = \mathsf{ID}[]$   $\mathsf{CTX}[\mathsf{cred}_\mathsf{m}] = "master"$ AND $\mathsf{OWNR}[\mathsf{cred}_\mathsf{m}]$
$\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], \mathsf{ctx}, aux), \mathsf{Issue}(\mathsf{osk}, \mathsf{ctx}, aux))$
Add $(\mathsf{cred}, i, \mathsf{ctx}) to (\mathsf{CRED}, \mathsf{OWNR}, \mathsf{CTX})$
return $\mathsf{cred}$
```

Fig. 15: V8 Note that $i$ indexes oracle calls, $j$ indexes issuer

### 18.10 Unforgeability Base Case Single Issuer V1

1. **Setup:**
   - $\mathsf{HU}, \mathsf{CU}$ are modelled as sets e.g. $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$ as sets prevent deuplicates
   - $\mathsf{CRED}$: each entry in $\mathsf{CRED}$ is a tuple $(\mathsf{cred}, \mathsf{cm}, m, i)$ where $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{opk})$ includes the signature $\sigma$ and commitment $\mathsf{cm}$, $m$ and $i$ track the message and user.
   - $\mathsf{OWNR}$: is a map supporting query $\mathsf{OWNR}[\mathsf{cred}] = i$
2. **Oracles**
   - $\mathcal{O}_{\mathsf{HU}}$ creates an honest user $i$ with secret key $\mathsf{usk}[i]$ modelling legitimate users
   - $\mathcal{O}_{\mathsf{CU}}$ corrupts an honest user $i$, giving adversary $\mathsf{usk}[i]$ and all credentials issued to $i$
   - $\mathcal{O}_{\mathsf{ObtIss}}$ issues a credential $\mathsf{cred}$ to an honest user $i$. Receives $m$ as input, computes commitment $\mathsf{cm} \leftarrow \mathsf{CM.Com}([m]; \mathsf{usk}[i])$ and $\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], ))$
3. **Adversary Goal:**
4. **Win Condition:** The adversary output must be valid, and can't have come from a corrupt user. That is, $\mathcal{A}$ must produce a valid, new output, not a trivial forgery, such as stealing someone's credentials, modeled by $\mathcal{O}_{\mathsf{CU}}$, which we reject in our win condition as well as $\mathcal{A}$ using a rerandomized version of the corrupt user credential.

```
Experiment $\mathrm{Exp}^{\mathsf{UNF}}_{\mathsf{SingleIssuer},\mathcal{A}}(\lambda)$
    // Challenger Setup
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{ck} \leftarrow \mathsf{CM.Setup}(1^\lambda)$
$(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{OrgKeyGen}(\mathsf{pp})$
$\{\mathsf{HU}, \mathsf{CU}, \mathsf{CRED}\} \leftarrow \emptyset, \mathsf{OWNR} \leftarrow \{\}$
    // Adversary Interaction
$\mathcal{A}^{\mathcal{O}_{\mathsf{HU}}, \mathcal{O}_{\mathsf{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{opk}, \mathsf{ck})$
$\mathcal{A}$ outputs $(\mathsf{cred}'^*, \mathsf{cm}'^*, \pi^*)$
    // Winning Condition
$\mathsf{Verify}(\mathsf{cred}'^*, \mathsf{cm}'^*, \mathsf{opk}, \pi^*) = 1$
$\wedge\ \forall \mathsf{cred}$ where $\mathsf{OWNR}[\mathsf{cred}] \in \mathsf{CU}$,
$\mathsf{cred}'^*$ is not a rerandomization of $\mathsf{cred}$

$\mathcal{O}_{\mathsf{HU}}(i)$
if $i \notin \mathsf{HU}$
    $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$
    $\mathsf{usk}[i] \leftarrow\$ \mathsf{UserKeyGen}(1^\lambda)$
return $i$

$\mathcal{O}_{\mathsf{CU}}(i)$
if $i \in \mathsf{HU} \wedge i \notin \mathsf{CU}$
    $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$
return $\mathsf{usk}[i]$,
$\{(\mathsf{cred}, \mathsf{cm}, m, i) \in \mathsf{CRED} \mid \mathsf{OWNR}[\mathsf{cred}] = i\}$

$\mathcal{O}_{\mathsf{ObtIss}}(i, m)$
if $i \notin \mathsf{HU}$, return $\perp$
    $\mathsf{cm} \leftarrow \mathsf{CM.Com}([m]; \mathsf{usk}[i])$
    $\mathsf{cred} \leftarrow (\mathsf{Obtain}(\mathsf{usk}[i], \mathsf{cm}), \mathsf{Issue}(\mathsf{osk}, \mathsf{cm}))$
    $\mathsf{CRED} \leftarrow \mathsf{CRED} \cup \{(\mathsf{cred}, \mathsf{cm}, m, i)\}$
    $\mathsf{OWNR}[\mathsf{cred}] \leftarrow i$
return $\mathsf{cred}$

$\mathcal{O}_{\mathsf{Show}}(i)$
if $i \notin \mathsf{HU}$, return $\perp$
    Select $(\mathsf{cred}, \mathsf{cm}, m, i) \in \mathsf{CRED}$
    $(\mathsf{cred}', \mathsf{cm}', \pi) \leftarrow \mathsf{Show}(\mathsf{cred})$
return $(\mathsf{cred}', \mathsf{cm}', \pi)$
```

## 18.11 Unforgeability Single Issuer Reduction

An adversary attempts to forge the signature, the commitment, or the credential. We want to show our system is secure by linking our primitives - signature, commitment, and zero-knowledge proofs, to underlying primitives.

We will assume A can break the security and forge a credential; then we construct a simulator B that uses A to break the security.

**Theorem 7 (Single Issuer Unforgeability).** *The single-issuer anonymous credential system is unforgeable, that is, it is secure against any adversary producing a credential that is not a rerandomized credential that has been legitimately issued and rerandomized by an honest, uncorrupt user, under the following assumptions*

1. *The underlying rerandomizable signature scheme is existentially unforgeable under chosen message attack* (EUF-CMA)
2. *the commitment scheme is computationally binding*
3. *the zero-knowledge proof system is computationally sound*

*Proof.* Assume there exists an adversary $\mathcal{A}$ that can break the unforgeability of the single-issuer anonymous credential system with non-negligible probability $\epsilon$. That is, $\mathcal{A}$ can produce a forged credential $\mathsf{cred}'^* = (\sigma'^*, \mathsf{cm}'^*, \mathsf{opk})$ and proof $\pi^*$ such that:

- $\mathsf{Verify}(\mathsf{cred}'^*, \mathsf{cm}'^*, \mathsf{opk}, \pi^*) = 1$ and
- $\mathsf{cred}'^*$ is not a rerandomization of a credential issued to a corrupt user

Then we construct simulators $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ to reduce $\mathcal{A}'s$ advantage to breaking one of the underlying assumptions

**Simulator $\mathcal{B}_1$, Reduction for EUF-CMA** If $\mathcal{A}$ can forge a credential, we can use $\mathcal{A}$ to break the EUF-CMA of the rerandomizable signature scheme. We construct Simulator $\mathcal{B}_1$ as follows:

- $\mathcal{C}$ sets up the reduction with $\mathsf{sk}, \mathsf{pk} \leftarrow\$ \mathsf{RS.KeyGen}()$ and oracle $\mathcal{O}_{\mathsf{sign}}(\mathsf{sk}, \cdot)$
- $\mathcal{B}_1$ receives $\mathsf{pk}$ from $\mathcal{C}$, sets $\mathsf{opk} = \mathsf{pk}$
- $\mathcal{B}_1$ simulates the credential system for $\mathcal{A}$
  - $\mathcal{B}_1$ simulates $\mathcal{O}_{\mathsf{Obtlss}}$. $\mathcal{A}$ provides $m$. $\mathcal{B}_1$ computes $\mathsf{cm} = \mathsf{CM.Com}([m]; r)$, queries oracle for $\sigma \leftarrow \mathcal{O}_{\mathsf{sign}}(\mathsf{cm})$. Returns $\mathsf{cred} = (\sigma, \mathsf{cm}, \mathsf{opk})$
  - for corruption queries, $\mathcal{B}_1$ reveals the internal state of corrupt users including $m, r, \sigma, etc$
- $\mathcal{A}$ outputs a forgery $\mathsf{cred}'^* = (\sigma'^*, \mathsf{cm}'^*, \mathsf{opk})$ and proof $\pi^*$ such that $\mathsf{Verify}(\mathsf{cred}'^*, \pi^*) = 1$ where $\mathsf{cred}'^*$ is not a rerandomized credential from a corrupt user, $\mathsf{cm}'^*$ was not signed by the oracle
- $\mathcal{B}_1$ outputs $\sigma'^*, \mathsf{cm}'^*$ as a forgery for the signature scheme

If $\mathcal{A}$ succeeds with probability $\epsilon$, then $\mathcal{B}_1$ breaks EUF-CMA with probability $\epsilon$, contradicting EUF-CMA security.

**Simulator $\mathcal{B}_2$** Our goal is to show that if $\mathcal{A}$ can produce a proof that verifies but hasn't been created from a valid credential, we can use $\mathcal{A}$ to break the soundness of the zero-knowledge proof system.

As for $\mathcal{B}_1$, $\mathcal{B}_2$ simulates the credential system, and when $\mathcal{A}$ outputs the forgery $(\mathsf{cred}'^*, \pi^*)$, $\mathcal{B}_2$ checks if $\pi^*$ verifies but the underlying statement is false:

- The statement $\exists\, m, r | \mathsf{cm}'^* = \mathsf{CM.Com}([m]; r)$ and $\sigma'^*$ is a valid signature on $\mathsf{cm}'^*$
- If $\pi^*$ verifies but the statement is false e.g. $\sigma'^*$ is invalid or $\mathsf{cm}'^*$ doesn't correspond to a valid $m'$, then $\pi^*$ is a false proof
- $\mathcal{B}_2$ outputs $\pi^*$ as a sound-breaking proof

If $\mathcal{A}$ succeeds with probability $\epsilon$, then $\mathcal{B}_2$ breaks proof soundness with probability $\epsilon$ contradicting the soundness of the proof system.

**Simulator $\mathcal{B}_3$** The goal is to show that if $\mathcal{A}$ can open $\mathsf{cm}'^*$ to different messages or attributes, we can use $\mathcal{A}$ to break the binding property of the commitment scheme. Simulator $\mathcal{B}_3$ works as follows:

- $\mathcal{B}_3$ simulates the credential system as before
- Suppose $\mathcal{A}$'s proof $\pi^*$ implies a message $m'$, different from $m$ within the commitment $\mathsf{CM.Com}([m]; r)$.
- We show that by using the soundness extractor $\mathcal{E}$, $\mathcal{B}_3$ obtains $(m, r), (m', r')$ for $\mathsf{cm}'^*$ where $m \neq m'$j
- $\mathcal{B}_3$ outputs $(m, r, m', r')$ that violates the commitment binding

If $\mathcal{A}$ succeeds with probability $\epsilon$ then $\mathcal{B}_3$ breaks commitment binding with probability $\epsilon$ contradicting the binding security of the commitment scheme.

## References

ASM06. M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. *Security and Cryptography for Networks*, 4116:111–125, 2006. Series Title: Lecture Notes in Computer Science.

BB04. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology - EUROCRYPT 2004*, pages 223–238, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

BB08. D. Boneh and X. Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

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.

BCD$^+$17. F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov. Accumulators with Applications to Anonymity-Preserving Revocation, 2017. Publication info: Published elsewhere. Minor revision. IEEE European Symposium on Security and Privacy 2017.

BCN$^+$10. P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get Shorty via Group Signatures without Encryption. In *Security and Cryptography for Networks*, pages 381–398, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

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.

BS23. M. Babel and J. Sedlmeir. Bringing data minimization to digital wallets at scale with general-purpose zero-knowledge proofs, January 2023. arXiv:2301.00823 [cs].

CDH16. J. Camenisch, M. Drijvers, and J. Hajny. Scalable Revocation Scheme for Anonymous Credentials Based on n-times Unlinkable Proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 123–133, Vienna Austria, October 2016. ACM.

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.

CDR16. J. Camenisch, M. Dubovitskaya, and A. Rial. UC Commitments for Modular Protocol Design and Applications to Revocation and Attribute Tokens. In *Advances in Cryptology – CRYPTO 2016*, pages 208–239. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. 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.

Cha85. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

CKS24. E. Crites, A. Kiayias, and A. Sarencheh. SyRA: Sybil-Resilient Anonymous Signatures with Applications to Decentralized Identity, 2024. Publication info: Preprint.

CL01. J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology — EUROCRYPT 2001*, pages 93–118. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. Series Title: Lecture Notes in Computer Science.

CL02. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology — CRYPTO 2002*, pages 61–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. Series Title: Lecture Notes in Computer Science.

CL03. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. *Security in Communication Networks*, 2576:268–289, 2003. Series Title: Lecture Notes in Computer Science.

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

Elt24. O. E. O. Eltayeb. The Crucial Significance of Governance, Risk and Compliance in Identity and Access Management. *Journal of Ecohumanism*, 3(4):2395–2405, August 2024.

FHS19. G. Fuchsbauer, C. Hanser, and D. Slamanig. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

LRSW00. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym Systems. In *Selected Areas in Cryptography*, pages 184–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000. Series Title: Lecture Notes in Computer Science.

MMZ$^+$21. 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. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1348–1366. IEEE, 2021.

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.

noa21. Happy 10th Birthday – AWS Identity and Access Management | AWS News Blog, May 2021. Section: Launch.

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.

PCB$^+$. R. Pang, R. Caceres, M. Burrows, Z. Chen, P. Dave, N. Germer, A. Golynski, K. Graney, N. Kang, L. Kissner, J. L. Korn, A. Parmar, C. D. Richards, and M. Wang. Zanzibar: Google's Consistent, Global Authorization System.

PS16. D. Pointcheval and O. Sanders. Short Randomizable Signatures. *Topics in Cryptology - CT-RSA 2016*, 9610:111–126, 2016. Series Title: Lecture Notes in Computer Science.

RARM. R. Rabaninejad, B. Abdolmaleki, S. Ramacher, and A. Michalas. Attribute-Based Threshold Issuance Anonymous Counting Tokens and Its Application to Sybil-Resistant Self-Sovereign Identity.

RPX$^+$22. D. Rathee, G. V. Policharla, T. Xie, R. Cottone, and D. Song. ZEBRA: SNARK-based Anonymous Credentials for Practical, Private and Accountable On-chain Access Control, 2022. Publication info: Preprint.

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).

SNA21. R. Soltani, U. T. Nguyen, and A. An. A Survey of Self-Sovereign Identity Ecosystem. *Security and Communication Networks*, 2021:1–26, July 2021. arXiv:2111.02003 [cs].

TBA$^+$22. A. Tomescu, A. Bhat, B. Applebaum, I. Abraham, G. Gueta, B. Pinkas, and A. Yanai. Utt: Decentralized ecash with accountable privacy. *Cryptology ePrint Archive*, 2022.

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 *Proceedings of the 39th Annual Computer Security Applications Conference*, pages 216–228, 2023.

ZYD$^+$22. X. Zhang, M. M. Yadollahi, S. Dadkhah, H. Isah, D. P. Le, and A. A. Ghorbani. Data breach: analysis, countermeasures and challenges. *International Journal of Information and Computer Security*, 19(3/4):402, 2022.