Anonymous Self-Credentials and their Application to Single-Sign-On

Jayamine Alupotha
University of Bern

Mariarosaria Barbaraci
University of Bern

Ioannis Kaklamanis
Yale University

Abhimanyu Rawat
Universitat Pompeu Fabra

Christian Cachin
University of Bern

Fan Zhang
Yale University

4 April 2025

Abstract

Modern life makes having a digital identity no longer optional, whether one needs to manage a bank account or subscribe to a newspaper. As the number of online services increases, it is fundamental to safeguard user privacy and equip service providers (SP) with mechanisms enforcing Sybil resistance, i.e., preventing a single entity from showing as many.

Current approaches, such as anonymous credentials and self-sovereign identities, typically rely on identity providers or identity registries trusted not to track users' activities. However, this assumption of trust is no longer appropriate in a world where user data is considered a valuable asset.

To address this challenge, we introduce a new cryptographic notion, Anonymous Self-Credentials (ASC) along with two implementations. This approach enables users to maintain their privacy within an anonymity set while allowing SPs to obtain Sybil resistance. Then, we present a User-issued Unlinkable Single Sign-On (U2SSO) implemented from ASC that solely relies on an identity registry to immutably store identities. A U2SSO solution allows users to generate unlinkable child credentials for each SP using only one set of master credentials. We demonstrate the practicality and efficiency of our U2SSO solution by providing a complete proof-of-concept.

1 Introduction

Digital identities have become essential in today's interconnected world. As nearly every aspect of human life has transitioned to a digital ecosystem, countless service providers (SPs) rely on these identities, making an online presence unavoidable. However, to ensure a reliable and fair service, SPs must safeguard against Sybil attacks [70], i.e., prevent users from creating multiple identities to exploit, abuse, and manipulate offered services. Ensuring the uniqueness and legitimacy of users has become of paramount importance, especially for e-commerce and social networks.

In the strictest sense, fulfilling these properties means mapping a user one-to-one to an identity, producing what we can call a *Sybil-resistant identity* [70]. Usually, SPs tie an identity to an email address or a phone number, which, although not providing complete Sybil resistance, are a scarce resource to obtain. The assumption is that creating multiple email addresses or phone numbers can be cumbersome and financially or administratively demanding for a single user. These Sybil-resistant identities enable an SP to allocate their limited resources more effectively, guaranteeing they will serve diverse users and prevent fraudulent users from monopolizing.

On the other hand, users who use the same email or phone number to sign up for different services create a clear link between their profiles. User tracking [6, 79] is a well-known issue in modern systems, as user data has become a high-value asset, often infringing on people's privacy in unexpected ways. Thus, a privacy-preserving sign-on mechanism that offers *unlinkability* across services has many advantages.

Sybil resistance and unlinkability appear contradictory: the first one encourages users to maintain a single identity, while the second advocates multiple unlinkable identities across different services.

Initial work focused more on privacy-enhanced membership rather than Sybil resistance, including anonymous credentials [28, 24], group signatures [29, 62], one-out-of-many proofs [50, 19], accumulators [78], and ring signatures [71, 93].

For instance, anonymous credentials [28, 24], also known as attribute-based credentials [59], function as follows: a user registers a master identity with an identity provider (IdP). When the user wants to create a pseudonym for an SP, the user sends a blinded pseudonym to the IdP, which then proves to the SP that the user is registered without seeing the pseudonym.

IdP-managed group signatures [29, 62], one-out-of-many proofs [50, 19, 49], accumulators [90, 78], and ring signatures [71, 93], all follow a similar approach. In these systems, the IdP or the group manager maintains a master identity list of legitimate users, known as the anonymity set. Users registering a pseudonym with an SP prove ownership of a master identity in the anonymity set.

However, these solutions are limited to membership assurance and do not provide Sybil resistance, as they do not prevent users from presenting multiple pseudonyms for an SP.

Despite this limitation, these solutions laid the groundwork for the privacy objectives [59] of credential systems: (1) *issuer unlinkability*, which prevents the IdP from linking the master identity to the pseudonym even if the IdP and SPs collude, and (2) *multi-verifier unlinkability*, which allows users to create different pseudonyms using the same master identity without revealing any connection.

A more recent line of work [64, 33, 69] solely focuses on introducing Sybil resistance along with unlinkability using the IdP with advanced cryptographic techniques. For example, a user with a master identity can get two proofs from the IdP to anonymously register two pseudonyms for the same SP. However, the proofs will be linkable, indicating that two pseudonyms were generated from the same master identity, allowing the SP to reject the second pseudonym and obtain Sybil resistance. Their major drawback is the reliance on a trusted IdP. This dependency raises two significant issues for online services.

First, users are required to obtain their identities from designated IdPs, which forces everyone to get their master identities from a limited number of trusted providers, centering the authority and workload around these IdPs. For instance, IdPs have the authority to censor users. Moreover, an IdP may fail or collapse (even in a distributed IdP scenario [33, 86, 69, 73], the underlying threshold assumptions may fail). In that case, this IdP approach creates a single point of failure for all users and SPs relying on it.

The second issue involves side-channel leaks, especially *timing attacks*. Each time users register for a new service, they interact with both the IdP and the SP. The timestamps of these interactions could allow colluding IdPs and SPs to link users despite the cryptographic anonymity of the system.

In contrast to solutions with IdP, *self-sovereign identity solutions* [91, 13, 41, 61, 74, 46, 16, 32, 65] offer Sybil resistance by replacing the identity provider with an immutable Identity Registry (IdR), which can be established using a decentralized distributed ledger like blockchains. In this model, when users want to register for a service, they publish self-generated pseudonyms on the IdR and share these pseudonyms with SPs. The IdR ensures Sybil resistance by requiring payment or some form of real-world identity verification, preventing users from creating unlimited pseudonyms. Additionally, the IdR promotes unlinkability by allowing users to generate distinct pseudonyms for each SP.

The drawback here is that unlinkability depends on strong trust assumptions about the IdR, i.e., the IdR must not be able to trace pseudonyms. For instance, if a user uses the same traceable payment method for all their pseudonyms, the IdR could potentially link these pseudonyms together, despite any cryptographic measures in place. In practice, many self-sovereign identities [61, 68, 46] are based on blockchains, where payment tracing is publicly available, compromising privacy.

A potential solution is to publish only a master identity on the IdR instead of the pseudonyms. However, we need a cryptographic mechanism that allows users to *self-prove* that a pseudonym used for a service is generated by an individual who owns a master identity in the IdR without revealing the specific master identity. Importantly, each master identity should be limited to one pseudonym per service to fulfill Sybil resistance. At first glance, existing self-proving solutions such as traceable ring

signatures [43, 42, 20, 83], anonymous messaging [85], and double-blind proof of existence [2] seem to address some aspects of this self-proving credential system. However, it remains unclear how to achieve certain critical features like robustness, revocation, and usability with these approaches.

Our research problem focuses on formalizing and implementing a fully functional self-proving credential system by securely composing existing cryptographic techniques to guarantee multi-verifier unlinkability and Sybil resistance.

1.1 Our Contribution

In this paper, we introduce the notion of *Anonymous Self-Credentials* (ASC) as a building block for a user-centric and privacy-supporting credential system. ASC establishes a framework for a group of provers to authenticate themselves to colluding verifiers using master identities and targets the following properties:

- *Unforgeability*: The authentication can only be created by a prover who knows a secret key to a master identity.
- *Robustness*: Malicious provers who create identities resembling correct master identities cannot impede the authentication of honest provers.
- *Sybil resistance*: Any two authentications by the *same* prover to the *same* verifier will be linkable.
- *Anonymity*: Authentications conceal the master identity of a prover within the group, provided that at least two provers created identities honestly.
- *Multi-verifier unlinkability*: Any two authentications for two verifiers conceal whether they are from the same prover or two different provers, even if the verifiers collude, assuming that at least two provers are honest.

Also, we present two Zero-Knowledge Arguments (ZKA) to realize ASC: Common Reference String-based Anonymous Self-Credentials (CRS-ASC) and Structured Reference String-based Anonymous Self-Credentials (SRS-ASC).

More importantly, we demonstrate how to utilize ASC constructions for a *User-Issued Unlinkable Single Sign-On* (U2SSO) system. This system ensures users' unlinkability across various service providers while managing a single master Sybil-resistant identity self-issued by the user. To store master identities, it uses a public immutable IdR, which effectively replaces the traditional IdP. We provide a proof-of-concept implementation of U2SSO, including a user program and a sample SP interface, that uses an Ethereum smart contract as the IdR. Moreover, we conduct a performance evaluation of our implementation showing the practicality of the solution, compared with closely related work.

Next, we outline the key concepts behind ASC and U2SSO.

Anonymous Self-Credentials Consider an ASC system with a set of provers and verifiers, where each verifier is assigned a unique identifier known as *verifier identifier*. A prover wants to show eligibility to register with the correct credentials for a verifier's service. During setup, each prover generates a *master credential*, i.e. a pair of a public *master identity* and a *master secret key*. Thus, the set of master identities is publicly available for both provers and verifiers, which we call the *anonymity set*.

When a prover wants to register with a verifier, she first generates a *pseudonym*. The prover then sends the pseudonym and a zero-knowledge proof to the verifier. The proof asserts that the prover owns one of the master identities in the anonymity set, without revealing which one.

Following this protocol, a prover can potentially register multiple pseudonyms from a single master identity. However, to ensure Sybil resistance, a prover is limited to generating only one pseudonym per verifier. The ASC protocol uses *nullifiers* to link each master identity to a unique verifier identifier. Technically speaking, a master credential can create only one nullifier for any number of proofs with the

same verifier, even for different pseudonyms. However, a correctly generated master credential produces different unlinkable nullifiers for different verifiers.

During the registration process, the prover presents the nullifier along with a second zero-knowledge proof to validate its authenticity while keeping the master identity hidden (In our constructions, both zero-knowledge proofs are combined into a composite proof).

Each verifier maintains their own local list of nullifiers used to register its pseudonyms. A verifier accepts a pseudonym only if its corresponding nullifier is not present in that list. Once the pseudonym is accepted, the verifier adds the new nullifier to its list, preventing the nullifier's reuse. Since each master identity allows for only one nullifier per verifier, the prover can register only one pseudonym, thereby effectively establishing *Sybil resistance*.

User-issued Unlinkable Single Sign-On A U2SSO credential system offers the following algorithms: (1) registering a master identity with the IdR, (2) generating unlinkable child credentials derived from the master identity as a pseudonym, (3) registering the pseudonym with an SP, (4) authenticating to the SP using the registered pseudonym, and (5) revoking or updating a pseudonym.

At setup, a user registers a master identity with the IdR, which stores batches of master identities, referred as anonymity sets. Once an anonymity set is established, the user can register a pseudonym with an SP. The registration process employs the ASC protocol to prove ownership of one of the identities in the anonymity set, ensuring user privacy and Sybil resistance to the SPs.

Future authentications are conducted using the registered pseudonym, which removes the need for further interactions with the IdR. As a result, the U2SSO system only requires complex cryptographic proofs and IdR interactions for less frequent events such as child identity registration, revocation, or modification. This leads to a more efficient and practical SSO mechanism.

Furthermore, each service provider can select their preferred method of pseudonym authentication, making U2SSO highly adaptable for both Web 2.0 and Web 3.0, such as digital signature-based or password-based authentication.

We implement the credential system for each of the ASC construction, differentiating between SRS-U2SSO and CRS-U2SSO. The first one has a master identity of 96 bytes and a proof size of 328 bytes, for any size N of the anonymity set. The second has master identities of 33 bytes and nullifiers of 32 bytes, but the proof size is logarithmic in N, e.g., 4KB for $N = 1024$.

The rest of the paper is organized as follows: Section 2 introduces the related work, Section 3 describes the notation and preliminaries, Section 4 formalizes the novel notion of ASC and defines the expected security properties, and Section 5 presents ASC constructions. U2SSO is formally described in Section 6, while implementation and performance details are presented in Section 7. Section 8 discusses complementary features, provides a comparison to other SSO solutions, and concludes the paper.

2 Related work

The rising adoption of digital identities has made reliable user identification, as well as maintaining user privacy, increasingly critical. Sybil resistance [35], also known as unclonability [34, 23], aims to mitigate a single entity from appearing as multiple entities. However, while this might be good for service providers, typical methods to provide it completely lack privacy. As a result, this issue has received significant attention in cryptographic research. We present four categories of related work presenting the different models and solutions applicable to a credential system guaranteeing both Sybil resistance and unlinkability.

Membership and multi-show unlinkability with IdP. Initial research focuses on achieving membership with multi-show unlinkability [59], where a user with a master identity from a designated IdP can request attestations or proofs that demonstrate her legitimacy. However, each attestation remains unlinkable due to cryptographic techniques like blind signatures [27, 26, 60], rerandomizable signatures [81], or attribute-based signatures [17, 87, 7, 82]. In other words, given any two attestations, no one, not even

the IdP, can determine whether they belong to the same user or to different users. Examples include classic anonymous credentials [24, 12, 25, 11, 15].

In addition to obtaining attestations from an IdP, some approaches allow users with an IdP-issued master credential to *self-attest* their identity and still maintain multi-show unlinkability. With group signatures [29], for instance, master identities are public keys issued by a group manager. However, these systems typically incorporate a trapdoor that allows the group manager to trace the user. Another mechanism is k-attestations [7, 8], which allows the generation of up to k attestations before they become traceable. These membership assurances provide a *basic level* of Sybil resistance, by imposing restrictions on who can get a master identity from the IdP (or group manager). However, in this setting, nothing prevents users from generating multiple attestations from the same master identity and presenting them to the same verifier.

A similar level of Sybil resistance can be achieved even in the absence of IdP-issued master identities by allowing the IdP to select which user-generated master identities to include in an anonymity set. Then, users can self-prove their membership in the anonymity set using one-out-of-many proofs [50, 19, 49], accumulators [90, 78], or ring signatures [71, 93].

Sybil resistance and multi-verifier unlinkability with IdP. Unclonable group signatures [34, 23], Sybil-resistant anonymous signatures [33], and Single-Sign-On solutions [69, 92, 95, 64, 10] solve the issue of Sybil resistance by introducing identifiers for all verifiers (referred to as context [34, 33] or domain [95]). In these frameworks, a user who possesses a public key issued by an IdP can self-generate signatures and nullifiers, referred to as context-specific identifiers [33], for different verifiers. However, for the same verifier, the nullifier is fixed. Therefore, if a user attempts to authenticate multiple times with the same verifier, the verifier can reject the user's signatures after one or several attempts, thereby achieving Sybil resistance. In contrast, any two signature-nullifier pairs created for different verifiers ensure multi-verifier unlinkability, meaning that colluding IdPs and verifiers cannot determine whether these pairs belong to the same user or two users.

The self-attesting approach offers significant advantages over IdP attestation because it minimizes interactions with the IdP. Moreover, this method prevents colluding IdPs and verifiers from deanonymizing users through side-channel information.

However, such self-attestations, e.g., Single-Sign-Ons [69, 92, 95, 64, 10], still depend on a trusted IdP to issue valid master credentials to users. Although these IdPs can operate in distributed setups with honest-majority assumptions [33, 86, 69, 73], it remains uncertain whether they can be effectively maintained in a global setup that involves numerous users and services with high stakes.

Sybil resistance with self-sovereign identities. Attempts to mitigate the trust put on the IdP [36, 77] started with PGP keys [5, 47], continued with the open identity layer [56, 76, 39], and now are gaining even more interest with the formalization of self-sovereign identities (SSI) [66]. However, a challenge emerged concerning Sybil resistance, as users could easily create multiple identities on their own [36]. This issue can be *mitigated* with blockchains [75, 37], which can require users to pay for registering their identities, thereby allowing the ledger to function as a passive IdR (controlling just onboarding and storage of identities). Numerous standards [91, 9] and implementations [13, 41, 61, 74, 46, 16, 32] have emerged from this effort.

In some of these systems [74, 46, 61], users create a unique pseudonym for each verifier or context and pay to store the pseudonym in the IdR. Before accepting a pseudonym, the verifier checks its existence in the IdR.

The major drawback of SSI implementations is the IdR traceability. For example, payments on blockchains (used as IdR) are publicly traceable, i.e., linking pseudonyms published by the same user is possible. As a result, implementations [61, 68, 46] on blockchains like Ethereum [37] lack multi-verifier unlinkability despite the cryptographic measures taken. Although untraceable blockchains [3] exist, they remain impractical due to scalability limitations.

Traceable ring signatures. Traceable ring signatures [43] and follow-up work [42, 20, 83, 88] are special signatures that can be linked by a "tag" if the signatures are created by the same prover for that specific tag. Moreover, they offer an additional property named exculpability [88] where an honest prover cannot be framed for creating linked signatures that were not generated by himself.

We demonstrate how any protocol satisfying ASC notion, such as traceable ring signatures [43, 42, 20, 83, 88], anonymous messaging [85], and double-blind proof of existence [2], can be turned into a comprehensive privacy-preserving credential system like U2SSO.

3 Preliminaries

In this section, we first define the notation used throughout the paper. Then, we present the cryptographic primitives that are used in our constructions.

3.1 Notation

$\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ is a ring of modular integers in $[0, q-1]$ for modulus q. $Pr[A|B]$ denotes the probability of A given B. λ indicates the security parameter, and $\mathcal{A}$ is a probabilistic polynomial-time (p.p.t.) adversary. We use $\mathcal{A}(s)$ to imply that $\mathcal{A}$ is given s. Also, we say $\epsilon(\lambda)$ is a negligible function if for all $c \in \mathbb{N}$ there exists x such that $\epsilon(\lambda) < \frac{1}{\lambda^c}$ for all $\lambda \geq x$. Sometimes, we casually use "negligible" to denote $\epsilon(\lambda)$. W.l.o.g, with p.p.t we always refer to a probabilistic polynomial-time algorithm that runs in time polynomial in λ. $s \xleftarrow{\$} \mathbb{S}$ denotes that s is drawn uniformly at random from a set $\mathbb{S}$. We use $[a_i]_{i=1}^{L}$=$[a_1, .., a_L]$ to denote a L element array, and $[[a_{i,l}]_{l=1}^{L}]_{i=1}^{N}$ for a $(N \times L)$ matrix of L columns and N rows.

3.1.1 Hash-based Key Derivation

A Hash-based Key Derivation Function (HKDF) [63] takes a random secret input key material $r \in [0,1]^m$ with sufficient entropy, proportional to the security parameter, $m \geq 2\lambda$, along with a salt $s \in [0,1]^{m'}$ (which can be public and not random), and outputs a pseudo-random key string $t \in [0,1]^n$ as follows:

$$t \leftarrow \mathsf{HKDF}^{n}_{m,m'}(r, s) \text{ where } r \xleftarrow{\$} [0,1]^{m \geq 2\lambda}.$$

HKDF is deterministic function, meaning that the same output is produced when the same input key material and salt are used. Given a good source of input key material, i.e., a high entropy and a good randomness (Definition 6 of [63]), a HKDF can be utilized as a pseudo-random key generator.

Consequently, the HKDF outputs are indistinguishable from random values, even when the salt is known, thus, they are also unlinkable. This property is crucial for our U2SSO constructions (Section 7), as it allows us to generate multiple pseudonyms from the same master key without revealing any information about the input key material. We formally define the unlinkability of HKDF in Definition 1.

Definition 1 (Unlinkability of HKDF). A HKDF of $(n, m \geq 2\lambda, m')$ is unlinkable if

$$Pr\left[\begin{array}{lll|r} b = b' & \wedge & & r_0 \xleftarrow{\$} [0,1]^m; r_1 \xleftarrow{\$} [0,1]^m; \\ s_0 \neq s_1 & \wedge & & (s_0, s_1) \leftarrow \mathcal{A}; b \xleftarrow{\$} [0,1] \\ s_0 \in [0,1]^{m'} & \wedge & & t_0 := \mathsf{HKDF}^{n}_{m,m'}(r_0, s_0) \\ s_1 \in [0,1]^{m'} & & & t_1 := \mathsf{HKDF}^{n}_{m,m'}(r_b, s_1) \\ & & & b' \leftarrow \mathcal{A}(t_0, t_1, s_0, s_1) \end{array}\right] \leq \frac{1}{2} + \epsilon(\lambda).$$

Note that this definition is a simplified version of Definition 7 from [63], emphasizing unlinkability for two queries.

3.1.2 Generalized Commitments

A commitment conceals a value using a randomly selected blinding key. A prover can publish the commitment and later reveal its contents by providing both the hidden value and the blinding key. This process is referred to as "opening the commitment". These commitments carry a binding property, meaning it is either computational difficult or impossible (perfect binding) for the prover to find an alternative value and blinding key that would also successfully open the commitment. Hence, the term "commitment" is used.

In this work, we employ multi-value commitments, which allow a single commitment to hide and bind up to L values using a single blinding key. We denote such a multi-value commitment scheme as follows:

- $\mathsf{ComSetup}(\lambda) : (crs, \mathbb{Z}_q, \mathbb{S})$ $\triangleright$ outputs the common reference string crs, the set $\mathbb{Z}_q$ for values, and the blinding key set $\mathbb{S}$.
- $\mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}([v_i]_{i=1}^L, k) : C$ $\triangleright$ outputs the commitment C for a blinding key $k \in \mathbb{S}$ and a vector of L values $[v_i]_{i=1}^L \in \mathbb{Z}_q^L$.
- $\mathsf{ComOpn}_{crs}^{\mathbb{Z}_q,\mathbb{S}}(C, [v_i]_{i=1}^L, k) : 1/0$ $\triangleright$ outputs 1 if the opening of C is $([v_i]_{i=1}^L, k)$; otherwise, returns 0.

Sometimes, we use a special case of these commitments that do not commit any value but only a blinding key, similar to public keys of digital signatures, and we denote them as $C = \mathsf{Com}_{crs}^{\mathbb{S}}(k \xleftarrow{\$} \mathbb{S})$. We formally define the hiding and binding properties of these multi-value commitments in Definition 2 and Definition 3, respectively.

Definition 2 (Hiding commitments). Suppose that $\mathcal{A}(crs, q, \mathbb{S})$ picks and shares two different value vectors $([v_{0,i}]_{i=1}^L, [v_{1,i}]_{i=1}^L) \in \mathbb{Z}_q^{2\times L}$ with the challenger after receiving $(crs, \mathbb{Z}_q, \mathbb{S}) := \mathsf{ComSetup}(\lambda)$. Then, the challenger shares C for a random choice of $b \xleftarrow{\$} [0, 1]$ where $C = \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}([v_{b,i}]_{i=1}^L, k \xleftarrow{\$} \mathbb{S})$. Multi-value commitments are hiding if the probability of $\mathcal{A}$ finding b is $\leq 1/2 + \epsilon(\lambda)$.

Definition 3 (Binding commitments). The commitment scheme of $(crs, \mathbb{Z}_q, \mathbb{S}) := \mathsf{ComSetup}(\lambda)$ is binding if the probability of $\mathcal{A}$ finding two different openings: $([v_i]_{i=1}^L, k) \in (\mathbb{Z}_q^L, \mathbb{S})$ and $([v'_i]_{i=1}^L, k') \in (\mathbb{Z}_q^L, \mathbb{S})$ such that $\mathsf{ComOpn}_{crs}^{\mathbb{Z}_q,\mathbb{S}}(C, [v_i]_{i=1}^L, k) = 1 \wedge \mathsf{Opn}_{crs}^{\mathbb{Z}_q,\mathbb{S}}(C, [v'_i]_{i=1}^L, k') = 1$ is negligible.

3.1.3 Zero-Knowledge Argument (ZKA)

We define the zero-knowledge argument (used [21, 67, 50]) for a polynomial time decidable relation $\mathcal{R}$ and three p.p.t. entities: a Common Reference String (CRS) generator Setup, a prover $\mathcal{P}$, and verifier $\mathcal{V}$. In some cases, the CRS has a special structure, and we refer to it as Structured Reference String (SRS). An SRS is a CRS generated from a complex distribution, typically using a sampling algorithm with internal randomness that must remain hidden to prevent the creation of fraudulent proofs.

In the context of ZKA, $\mathcal{P}$ demonstrates a statement u regarding a witness w without disclosing any additional information about w. We denote the relation $\mathcal{R}$ for the generated crs as: $(crs, u, w) \in \mathcal{R}$. For instance, suppose that $\mathcal{P}$ proves that some commitment $C_j \in [C_1, ..., C_N]$ commits zero by generating a proof π without revealing its index j. Such relation $\mathcal{R}_{zero,N}$ is denoted as follows:

$$\begin{aligned} \mathcal{R}_{zero,N} : \big(crs, u := ([C_i]_{i=1}^N, \pi), w := (j, k)\big) \quad \Leftrightarrow \\ \Big(C_j = \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}(0 \in \mathbb{Z}_q, k \in \mathbb{S}) \wedge j \in [1, N]\Big). \end{aligned} \tag{1}$$

Here, the witness consists of the index j and the key k of commitment C_j. The statement u contains the proof π and the set $[C_i]_{i=1}^N$.

We call all statements that have witness(es) for crs a CRS-dependent language $\mathcal{L}_{crs} = \{x | \exists w : (crs, x, w) \in \mathcal{R}\}$. A proving algorithm may take multiple interactions between $\mathcal{P}$ and $\mathcal{V}$. We denote an interaction's transcript as $tr \leftarrow \langle \mathcal{P}(crs, u, w), \mathcal{V}(crs, u) \rangle$. If $\mathcal{V}$ accepts tr, then $\mathcal{V}(tr) = 1$.

Definition 4 (Zero-Knowledge Argument). $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a ZKA for relation $\mathcal{R}$ if they satisfy the following properties:

- **Completeness**: Let $\mathcal{B}$ be a p.p.t. adversary who generates witness w and statement u. $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is complete if

$$Pr\left[\begin{matrix}(crs,u,w)\overset{?}{\notin}\mathcal{R} \quad \vee \\ \mathcal{V}(\langle\mathcal{P}(crs,u,w),\mathcal{V}(crs,u)\rangle)\overset{?}{=}1\end{matrix} \quad \middle| \quad (u,w)\leftarrow\mathcal{B}(crs)\right] = 1.$$

- **Computational Witness-Extended Emulation (WEE)**: Let there be two p.p.t. adversaries: WEE adversary $\mathcal{A}$ and $\mathcal{B}$ who generates the initial witness s and statement u. Also, $\mathcal{P}^*$ is a deterministic p.p.t. prover, and $\mathcal{E}^{\mathcal{W}}$ is a p.p.t. emulator that generates an emulated transcript $(tr, w) \leftarrow \mathcal{E}^{\mathcal{W}}(u)$ with a witness w such that $\mathcal{V}(tr) = 1$. $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ has WEE if:

$$\left|\begin{matrix}Pr\left[\mathcal{A}(tr)\overset{?}{=}1 \quad \middle| \quad \begin{matrix}crs:=\mathtt{Setup}(\lambda),(u,s)\leftarrow\mathcal{B}(crs)\\ tr\leftarrow\langle\mathcal{P}^*(crs,u,s),\mathcal{V}(crs,u)\rangle\end{matrix}\right] \quad - \\ Pr\left[\begin{matrix}\mathcal{A}(tr)\overset{?}{=}1 \quad \wedge \\ (\mathcal{V}(tr)\overset{?}{\Rightarrow}(crs,u,w)\in\mathcal{R})\end{matrix} \quad \middle| \quad \begin{matrix}crs:=\mathtt{Setup}(\lambda),(u,s)\leftarrow\mathcal{B}(crs)\\ (tr,w)\leftarrow\mathcal{E}^{\mathcal{W}(\langle\mathcal{P}^*(crs,u,s),\mathcal{V}(crs,u)\rangle)}(crs,u)\end{matrix}\right]\end{matrix}\right| \leq \epsilon(\lambda)$$

  If $\mathcal{A}$ cannot identify emulated transcripts over genuine transcripts, i.e., $\mathcal{A}$ accepts emulated transcripts $\mathcal{A}(tr) = 1$, it implies that $\mathcal{A}$ learns nothing about w, except $\mathcal{R}$.

- **Knowledge Soundness**: Let $\mathcal{P}^*$ be a rewindable prover and $\mathcal{W}$ be a p.p.t. extractor which extracts witnesses by rewinding $\mathcal{P}^*$ to a certain iteration of witness s and resuming with fresh verifier randomness, i.e., $w \leftarrow \mathcal{W}(\langle\mathcal{P}^*(crs,u,s),\mathcal{V}(crs,u)\rangle)$. The previous negligible function of WEE also captures the knowledge soundness if $\mathcal{P}^*$ cannot generate valid transcripts that do not align with the relation, i.e., $(\mathcal{V}(tr)\overset{?}{=}1 \wedge (crs,u,w)\overset{?}{\notin}\mathcal{R})$.

Multi-Prover ZKA Setting In this paper, we focus on multi-prover protocols in which a subset of provers may be compromised by an adversary during the WEE (Definition 4). We denote the indices of the honest provers in the WEE by $J_{WEE} := [j_1, \ldots, j_{N-N'}]$ where N' is the number of corrupted provers. Thus, the relation $\mathcal{R}_{zero,N}$ (Eq. (1)) can be rewritten as follows:

$$\begin{aligned}\mathcal{R}_{zero,N,J_{WEE}} : \big(crs, u := ([C_i]_{i=1}^N, \pi), w := (j,k)\big) \quad &\Leftrightarrow \\ \Big(C_j = \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}(0 \in \mathbb{Z}_q, k \in \mathbb{S}) \wedge j \in J_{WEE} \wedge |J_{WEE}| \geq 2\Big)&.\end{aligned}$$

The zero-knowledge property of the witness holds only if the commitment j belongs to an honest prover such that $j \in J_{WEE}$ and there exists at least one more honest prover: $|J_{WEE}| \geq 2$.

Non-Interactive ZKA Many of our protocols are non-interactive, i.e., the prover directly sends the entire proof to the verifier without any intermediate interactions. We define such non-interactive argument system in Definition 5, following [21].

Definition 5 (Perfect Special Honest-Verifier Zero-Knowledge). A public coin argument of knowledge $(\mathtt{Setup}, \mathcal{P}, \mathcal{V})$ is a perfect special honest verifier zero-knowledge argument (SHVZK) of knowledge for $\mathcal{R}$ if there exists a p.p.t. simulator $\mathcal{S}$ such that for all pairs of interactive adversaries $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{aligned}&Pr\left[\begin{matrix}(crs,u,w)\in\mathcal{R} \\ \wedge \quad \mathcal{A}(tr)=1\end{matrix} \quad \middle| \quad \begin{matrix}crs := \mathtt{Setup}(\lambda); (u,w,\rho)\leftarrow\mathcal{B}(crs)\\ tr\leftarrow\langle\mathcal{P}(crs,u,w),\mathcal{V}(crs,u;\rho)\rangle\end{matrix}\right] \quad = \\ &Pr\left[\begin{matrix}(crs,u,w)\in\mathcal{R} \\ \wedge \quad \mathcal{A}(tr)=1\end{matrix} \quad \middle| \quad \begin{matrix}crs := \mathtt{Setup}(\lambda)\\ (u,w,\rho)\leftarrow\mathcal{B}(crs); tr\leftarrow\mathcal{S}(u,\rho)\end{matrix}\right]\end{aligned}$$

where ρ is the public randomness used by the verifier.

3.1.4 Strong Unforgeability for a Generic Prover-Verifier Setup

In this paper, we define strong unforgeability in a generalized manner to apply for any generic prover and verifier. Such unforgeability is required when the prover wants to send an additional message, referred to as the "initial message", along with the proof to the verifier. Although this message is not related to the ZKA statement being proven, the prover aims to bind the proof's challenge to this initial message. Consequently, a proof with a fresh message can only be generated if the witness is known[1]. In this way, the verifier can authenticate that the initial message came from a prover who knows a witness to the statement and that it was not modified during a man-in-the-middle attack. We denote a proof created for an initial message M as $\pi(M) \in u$. Adopting this notation, we present the generic strong unforgeability in Definition 6.

Definition 6. Let there be a ZKA $(\mathtt{Setup}, \mathcal{P}, \mathcal{V})$ where a proof $\pi(M) \in u$ can be created for any $M \in \mathbb{M}$, if $(crs, u, w) \in \mathcal{R}$. Let $\mathcal{W}(crs)$ be a p.p.t. witness picking algorithm that randomly picks a witness w such that there exists at least one possible statement u such that $(crs, u, w) \in \mathcal{R}$. Also, $\mathsf{Com}(w)$ denotes the hiding commitment of w. Suppose that there exists an oracle $\mathcal{O}_{\mathcal{P}(crs,w)}(M)$ that generates statements $u(M)$ for any given query M using $\mathcal{P}$ and stores the queries such that $\mathbf{Q} := \mathbf{Q} \cup \{M\}$. The ZKA system provides strong unforgeability for any p.p.t. $\mathcal{A}^{\mathcal{O}_{\mathcal{P}(crs,w)}(\cdot)}$ that has access to this oracle if

$$Pr\left[\begin{array}{l|r} M' \notin \mathbf{Q} \wedge \pi(M') \in u \quad \wedge & crs := \mathtt{Setup}(\lambda) \\ (crs, u, w) \in \mathcal{R} & w \xleftarrow{\$} \mathcal{W}(crs); c := \mathsf{Com}(w) \\ & (u, M') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{P}(crs,w)}(\cdot)}(crs, c) \end{array}\right] \leq \epsilon(\lambda).$$

The ASC constructions used in U2SSO (Section 7) obtain strong unforgeability via hashing the initial message during the Fiat-Shamir transformation [40] to generate non-interactive challenges.

4 Anonymous Self-Credentials (ASC)

In this section, we describe ASC and their security properties, including Sybil resistance and multi-verifier unlinkability.

4.1 System Model

An anonymous self-credential is a protocol encompassing N provers and L verifiers. Each verifier $l \in [1, L]$, is assigned a unique identifier called a *verifier identifier*, represented as $v_l \in \mathbb{V}$ from some predefined set $\mathbb{V}$.

Each prover $j \in [1, N]$ possesses a master credential (Φ_j, sk_j), which consists of a public master identity Φ_j and a master secret key sk_j. The collection of master identities from all provers forms what is known as the anonymity set, represented as $\Lambda := [\Phi_i]_{i=1}^{N}$. During the setup phase, all verifiers and provers are informed about the list of verifier identifiers and the anonymity set.

To access the services provided by verifiers, provers must first complete a registration process. Instead of using their master credentials for registration, provers utilize a pseudonym ϕ (which can be a username or any public information related to another credential) to protect their privacy. The use of ASC does not restrict the method of generating pseudonyms; for example, the format of the pseudonym can be determined by the verifier. To frame this in a cryptographic context, we define a generic interface for a pseudonyms' generator $\mathcal{G}$ (Section 6).

The primary objective of this scheme is to ensure Sybil resistance, preventing any prover from registering multiple pseudonyms with the same verifier. To accomplish this, the scheme employs nullifiers. If

[1]Mon-malleability of ZKA states that a fresh proof cannot be generated without knowing the witness [44]. Hence, strong unforgeability differs from non-malleability since unforgeability is about creating a proof for a fresh message where the proof does not have to be fresh. Additionally, it is important to note that non-malleability is already encompassed by the concept of extractability as defined in Definition 4. For a more detailed explanation, readers can refer to [44].

Condition	Provers' State	Relation
$j = j' \wedge l = l'$ for any (ϕ, ϕ')	honest or malicious	$nul = nul'$
$j \neq j' \vee l \neq l'$ for any (ϕ, ϕ')	honest	$nul \neq nul'$

Table 1. Nullifiers for two registrations of (prover j, verifier l, pseudonym ϕ) and (prover j', verifier l', pseudonym ϕ').

a prover attempts to register more than one pseudonym with the same verifier, the nullifier will be identical, as illustrated in Table 1. It is important to note that the nullifiers do not depend on the pseudonym. Due to these decoupled pseudonyms, ASC can be integrated with any pseudonym format preferred by the verifier, including signature or password-based credentials.

To register a pseudonym, a prover generates a proof π along with a nullifier nul. This proof π serves two purposes: first, it demonstrates that the pseudonym was generated by a prover who possesses a master secret key within the anonymity set; second, it confirms that the nullifier was correctly generated from the *same* master credential for the verifier l. The prover then sends (ϕ, nul, π) to the verifier. Each verifier l has a *local list of nullifiers* that have been used to register all previous pseudonyms for verifier l. Then the verifier l checks the validity of the proof and ensures that the nullifier has not been used before. If both conditions are satisfied, the verifier accepts the pseudonym for registration.

Additionally, master credentials that are generated honestly produce different nullifiers for each verifier in order to maintain correctness (see Table 1). This allows an honest prover to register with any verifier successfully.

4.2 Protocol

We formally define ASC functionalities below:

- $\mathsf{ASC.Setup}(\lambda, L) \rightarrow (crs) \triangleright$ Generates crs, including the list of verifier identifiers $[v_l]_{l=1}^{L} \in crs$, given a security parameter λ and defines the key space $\mathbb{K}$ for master secret keys. Here, each verifier has a unique identifier.

- $\mathsf{ASC.Gen}(crs, sk \in \mathbb{K}) \rightarrow \Phi \triangleright$ A prover self-generates a master credential, which includes a master identity Φ from a master secret key $sk \in \mathbb{K}$.

- $\mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi) \rightarrow (nul, \pi) \triangleright$ The algorithm outputs a nullifier nul and a proof π to register a some pseudonym ϕ for verifier l, identified by $v_l \in crs$. This algorithm is used by a prover j with master credential represented by (Φ_j, sk_j), where Φ_j is in the anonymity set Λ, i.e., $\Phi_j \in \Lambda$.

- $\mathsf{ASC.Open}(crs, l, nul, \Phi, sk_j) \rightarrow 1/0 \triangleright$ A prover can later open and identify whether the nullifier was generated by herself using the knowledge of master secret key sk_j (this is not a function for verifiers).

- $\mathsf{ASC.Verify}(crs, \Lambda, l, \phi, nul, \pi) \rightarrow 1/0 \triangleright$ Anyone can verify that the pseudonym ϕ was generated by a prover who owns a master identity in the anonymity set Λ, and that the nullifier nul is correctly derived from the same master credential without learning which one.

4.2.1 Correctness

ASC enables any honest prover to generate a master credential. This master credential can then be used to register any pseudonym for verifiers while incorporating nullifiers. For correctness, we utilize an external generator $\mathcal{G}$, which produces pseudonyms, indicating that ASC is independent of how pseudonyms are generated.

Definition 7 (Correctness). Consider an anonymous self-credential protocol ASC with N provers and L verifiers. We define the correctness game, in which two honest provers (j, j') attempt to register two pseudonyms, (ϕ, ϕ') generated by any external generator $\mathcal{G}$, for two verifiers, (l, l') as follows:

$\mathsf{Game}^{Correct}_{\mathsf{ASC},\mathcal{G}}(\lambda, L, N, j, j', l, l')$:
$crs := \mathsf{ASC.Setup}(\lambda, L)$ such that $[v_i]_{i=0}^{L} \in crs$
$[\Phi_i := \mathsf{ASC.Gen}(crs, sk_i \xleftarrow{\$} \mathbb{K})]_{i=1}^{N}$ and $\Lambda := [\Phi_i]_{i=1}^{N}$ ▷ anonymity set
$(\phi, \phi') \xleftarrow{\$} \mathcal{G}$ ▷ get pseudonyms
$(nul, \pi) := \mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi)$
$(nul', \pi') := \mathsf{ASC.Prove}(crs, \Lambda, l', j', sk_{j'}, \phi')$
return $\mathsf{ASC.Verify}(crs, \Lambda, l, \phi, nul, \pi) \wedge \mathsf{ASC.Verify}(crs, \Lambda, l', \phi', nul', \pi')$
$\wedge \mathsf{ASC.Open}(crs, l, nul, \Phi_j, sk_j) \wedge \mathsf{ASC.Open}(crs, l', nul', \Phi_{j'}, sk_{j'})$
$\wedge\big((j = j' \wedge l = l' \wedge nul = nul') \vee ((j \neq j' \vee l \neq l') \wedge (nul \neq nul'))\big)$

We say that ASC is correct if

$$\forall L \in \mathbb{N}, N \in \mathbb{N}, (j, j') \in [1, N]^2, (l, l') \in [1, L]^2 :$$
$$Pr\left[\mathsf{Game}^{Correct}_{\mathsf{ASC},\mathcal{G}}(\lambda, L, N, j, j', l, l') \;\middle|\; N \geq 2\right] \geq 1 - \epsilon(\lambda).$$

4.3 Security

An anonymous self-credential system must guarantee the following security properties: robustness, unforgeability, Sybil resistance, anonymity, and multi-verifier unlinkability.

4.3.1 Robustness

During the registration process, the anonymity set may contain seemingly valid master identities that were not properly generated by the function $\mathsf{ASC.Gen}()$. However, these simulated master identities should not impact the correct registration of an honest prover. The fault tolerance of ASC is essential for real-world applications, as it ensures that provers who include incorrect master identities, intentionally or accidentally, cannot hinder the registration of honest provers. We formally define robustness of ASC in Definition 8.

Definition 8 (Robustness). Consider an anonymous self-credential protocol ASC with N provers and L verifiers. Among the provers, there is an honest prover j who generates the master credential correctly. The other provers obtain their master identities from $\mathcal{A}$, which outputs simulated master identities that resemble genuine master identities but may or may not have been generated correctly using $\mathsf{ASC.Gen}()$. We define the robustness game, in which the honest prover can still generate valid nullifiers and proofs as follows:

$\mathsf{Game}^{Robust}_{\mathsf{ASC},\mathcal{G},\mathcal{A}}(\lambda, L, N)$:
$crs := \mathsf{ASC.Setup}(\lambda, L)$
$j \xleftarrow{\$} [1, N]$ ▷ pick a prover
$\Phi_j := \mathsf{ASC.Gen}(crs, sk_j \xleftarrow{\$} \mathbb{K})$ ▷ honest prover's
$(l, [\Phi_i]_{i=1, i\neq j}^{N}) \leftarrow \mathcal{A}(crs)$ ▷ simulated master credentials and picked the verifier $l \in [1, L]$
$\Lambda := [\Phi_i]_{i=1}^{N}$ ▷ anonymity set
$\phi \xleftarrow{\$} \mathcal{G}$ ▷ get pseudonym
$(nul, \pi) := \mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi)$
return $\mathsf{ASC.Verify}(crs, \Lambda, l, \phi, nul, \pi)$

We say that ASC is robust if $\forall(L \in \mathbb{N}, N \in \mathbb{N})$:

$$Pr[\mathsf{Game}^{Robust}_{\mathsf{ASC},\mathcal{G},\mathcal{A}}(\lambda, L, N)] \geq 1 - \epsilon(\lambda).$$

4.3.2 Strong Unforgeability

ASC should satisfy a soundness property known as unforgeability. This means that an adversary who does not have access to the master secret key of an honest prover cannot generate a valid nullifier or proof.

In the context of ASC, it is important to consider a more robust adversarial model, where the adversary may compromise other provers (except for one honest prover) and select master identities for the anonymity set. However, even with this capability, the adversary should not be able to forge a valid proof or nullifier to register a pseudonym without the knowledge of the honest prover's master secret key.

We emphasize that ASC is only viable if strong unforgeability is provided. Thus, even if an adversary can request proofs for multiple pseudonyms of its choice from the honest prover, the adversary should not be able to use that information to forge a proof for a fresh pseudonym.

In the context of the protocol, a prover may share multiple ASC proofs for the same pseudonym on different occasions. This could occur due to interrupted communication by an adversary or for specific application purposes—for example, U2SSO uses ASC proofs for different functionalities like registration, revocation, and modification of pseudonyms. In such cases, the proofs may vary due to the internal randomness employed in the underlying constructions, even when the pseudonym remains the same. Strong unforgeability ensures that, despite these variations in proofs, the adversary cannot generate proof for a fresh pseudonym.

We define strong unforgeability in Definition 9.

Definition 9 (Strong Unforgeability). Let there be an anonymous self-credential protocol ASC with N provers and L verifiers. In the game of unforgeability, the adversary $\mathcal{A}$ is allowed to generate all master credentials on behalf of the provers, with the exception of one honest prover j. The objective of $\mathcal{A}$ is to forge a nullifier and a proof that are valid for any verifier l for any fresh pseudonyms ϕ', using the master credential of that honest prover. Here, a fresh pseudonym is a one that was not stored in $\mathbf{Q}$. We define this game as follows:

$\mathsf{Game}^{Unforge}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)$:
$crs := \mathsf{ASC.Setup}(\lambda, L)$ such that $[v_l]_{l=0}^{L} \in crs$
$\Phi := \mathsf{ASC.Gen}(crs, sk \xleftarrow{\$} \mathbb{K})$ ▷ honest master credential
$(j, [\Phi_i]_{i=1, i\neq j}^{N}) \leftarrow \mathcal{A}(crs, \Phi)$ ▷ adversary creates the rest
$\Lambda := [\Phi_i]_{i=1}^{N}$ when $(\Phi_j, sk_j) := (\Phi, sk)$ ▷ anonymity set
$\mathbf{Q} = \{\ \}$
While $(\phi, l) \leftarrow \mathcal{A}(crs, \Phi_j)$: ▷ allow multiple queries
  $(nul, \pi) := \mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi)$
  $(nul, \pi) \rightarrow \mathcal{A}$ and $\mathbf{Q} := \mathbf{Q} \cup \{\phi\}$
$(\phi', nul', \pi') \leftarrow \mathcal{A}(crs, \Phi_j)$
return $\phi' \notin \mathbf{Q} \wedge \mathsf{ASC.Verify}(crs, \Lambda, l, \phi', nul', \pi')$
$\wedge \mathsf{ASC.Open}(crs, l, nul', \Phi, sk_j)$

ASC is unforgeable if: for all $(L \in \mathbb{N}, N \in \mathbb{N})$:

$$Pr[\mathsf{Game}^{Unforge}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)] \leq \epsilon(\lambda).$$

4.3.3 Sybil Resistance

Sybil resistance of an ASC means that a master identity can only produce one nullifier for each verifier identifier.

Definition 10 (Sybil Resistance). Let ASC be an anonymous self-credential protocol with N provers and L verifiers. In the game of Sybil resistance, the adversary $\mathcal{A}$ can generate all master credentials on behalf of the provers. The goal of $\mathcal{A}$ is to find two tuples of pseudonyms, nullifiers, and proofs that are valid for the same verifier l and from the same master credential when the nullifiers are different.

We define this game as follows:

$\mathsf{Game}^{Sybil}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N) : crs := \mathsf{ASC.Setup}(\lambda, L)$
$(\Lambda, sk, l, (\phi, nul, \pi), (\phi', nul', \pi')) \leftarrow \mathcal{A}(crs)$
return $\mathsf{ASC.Verify}(crs, \Lambda, l, \phi, nul, \pi)$
$\wedge \mathsf{ASC.Verify}(crs, \Lambda, l, \phi', nul', \pi') \wedge nul \neq nul'$
$\wedge \mathsf{ASC.Open}(crs, l, nul, \Phi, sk) \wedge \mathsf{ASC.Open}(crs, l, nul', \Phi, sk)$

ASC offers Sybil resistance if: for all $(L \in \mathbb{N}, N \in \mathbb{N})$:

$$Pr[\mathsf{Game}^{Sybil}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)] \leq \epsilon(\lambda).$$

4.3.4 Anonymity

ASC systems should provide two key privacy features, the first of which is anonymity. Anonymity ensures that honest-but-curious verifiers cannot determine which master identity within the anonymity set conducted a registration.

Moreover, anonymous credentials maintain anonymity even under a strong adversarial model. In this context, an adversarial verifier may query a pseudonym's registration multiple times. For instance, in real-world scenarios, a malicious verifier might repeatedly claim not to have received the complete proof due to communication errors, allowing them to collect multiple proofs from the same prover (note that while the nullifier remains the same, the proofs may differ). Another example involves a malicious verifier rejecting a pseudonym and claiming that it has already been used. The verifier then requests different pseudonyms along with valid proof.

ASC must accommodate a maximum of N' malicious provers out of N total provers, provided that $N - N' \geq 2$. In this situation, an honest prover can still obscure their master identity among the remaining $N - N'$ honest provers.

Consequently, while the presence of malicious provers may reduce the size of the true anonymity set, it does not divulge any information about the master identity of the honest prover. This property is crucial for applications, as it is possible for certain provers to collude with verifiers. We define anonymity in Definition 11.

Definition 11 (Anonymity). Let ASC be an anonymous self-credential protocol, involving N provers and L verifiers. Among the N provers, N' provers may be malicious, with the condition that $N - N' \geq 2$. We define the index set of honest provers as $J := \{j_1, j_2, \ldots, j_{N-N'}\}$. While the honest provers correctly generate their master credentials, the adversary produces the master identities for the compromised provers in the index set $[i]^N_{i=1, i \notin J}$.

Thus, the anonymity set Λ comprises the master identities from both honest and compromised provers.

In the game, we randomly select an honest prover j such that $j \xleftarrow{\$} J$. The p.p.t. adversary can query multiple registrations from this honest prover for a verifier l for any pseudonym ϕ chosen by the adversary. The adversary's objective is to identify the index of the honest prover. We define the game of anonymity as follows:

$\mathsf{Game}^{Anonymity}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)$:
$crs := \mathsf{ASC.Setup}(\lambda, L)$ such that $[v_l]^L_{l=0} \in crs$
$(J, l) \leftarrow \mathcal{A}(crs)$
$[\Phi_i := \mathsf{ASC.Gen}(crs, sk_i \xleftarrow{\$} \mathbb{K})]_{i \in J}$
$[\Phi_i]^N_{i=1, i \notin J} \leftarrow \mathcal{A}(crs, [\Phi_{j_i}]_{i \in J})$ ▷ corrupted credentials
$\Lambda := [\Phi_i]^N_{i=1}$ ▷ anonymity set
$j \xleftarrow{\$} J$ ▷ pick a honest prover
While $\phi \leftarrow \mathcal{A}(crs, \Lambda)$ queries:
$(nul, \pi) := \mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi)$
$(nul, \pi) \rightarrow \mathcal{A}$ ▷ send the outputs to $\mathcal{A}$
$j' \leftarrow \mathcal{A}$ ▷ get the guessed index from $\mathcal{A}$
return $j = j' \wedge |J| \geq 2$

ASC provides the anonymity if for all $(L \in \mathbb{N}, N \in \mathbb{N})$:

$$Pr[\mathsf{Game}^{Anonymity}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)] \leq 1/(N - N') + \epsilon(\lambda).$$

4.3.5 Multi-Verifier Unlinkability

The second privacy feature of ASC is that registrations for two different verifiers should be unlinkable. Suppose verifier l receives a registration consisting of (ϕ, nul, π), and verifier l' receives a registration with (ϕ', nul', π'). Even if the two verifiers collude, they should not be able to determine whether the registrations came from the same master credentials or from two different ones. As a result, provers can register with multiple verifiers using the same master credential without allowing the colluding verifiers to link their registrations.

ASC should guarantee multi-verifier unlinkability when the anonymity set includes N' corrupted provers' master identities, provided that $N - N' \geq 2$. In other words, as long as there are at least two honest provers, the honest provers can maintain unlinkability despite the presence of malicious provers.

Definition 12 (Multi-verifier Unlinkability). Let ASC be an anonymous credential protocol, involving N provers and L verifiers. Among the N provers, up to N' may be malicious, with the condition that $N - N' \geq 2$. We define the index set of honest provers as $J := \{j_1, j_2, \ldots, j_{N-N'}\}$. While the honest provers correctly generate their master credentials, the adversary produces the master identities for the compromised provers in the index set $[i]^N_{i=1, i \notin J}$.

Consequently, the anonymity set Λ includes the master identities from both the honest and compromised provers.

First the adversary picks two pseudonyms (ϕ, ϕ'), and then the unlinkability game operates in two modes: (1) in the first mode two registrations are generated from the same master credential for verifier l and verifier l' when $l \neq l'$; (2) in the second mode two sets of registration data are produced from two different master credentials. The game randomly selects one of these modes and sends the corresponding data to the adversary. The adversary's goal is to determine which mode was used based on the registration data provided.

$\mathsf{Game}^{Unlink}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)$: $crs := \mathsf{ASC.Setup}(\lambda, L)$ such that $[v_l]^L_{l=0} \in crs$
$[\Phi_i := \mathsf{ASC.Gen}(crs, sk_i \xleftarrow{\$} \mathbb{K})]_{i \in J}$
$(l, l', [\Phi_i]^N_{i=1, i \notin J}) \leftarrow \mathcal{A}(crs, [\Phi_{j_i}]_{i \in J})$ ▷ corrupted credentials
$\Lambda := [\Phi_i]^N_{i=1}$ ▷ anonymity set
$(\phi, \phi') \leftarrow \mathcal{A}(crs)$ s.t. $\phi \neq \phi'$ ▷ different pseudonyms
$b \xleftarrow{\$} [0, 1]$ ▷ pick the mode
if $b = 0$: $j \xleftarrow{\$} J$ and $j' := j$ ▷ the mode of the same prover
if $b = 1$: $j \xleftarrow{\$} J$ and $j' \xleftarrow{\$} J \setminus [j]$ ▷ the mode of two different provers
$(nul, \pi) := \mathsf{ASC.Prove}(crs, \Lambda, l, j, sk_j, \phi)$
$(nul', \pi') := \mathsf{ASC.Prove}(crs, \Lambda, l', j', sk_{j'}, \phi')$
$b' \leftarrow \mathcal{A}((nul, \pi), (nul', \pi'))$; return $b = b' \wedge l \neq l'$

Anonymous self-credentials are multi-verifier unlinkable if for all $(L \in \mathbb{N}, N \in \mathbb{N})$:
$Pr[\mathsf{Game}^{Unlink}_{\mathsf{ASC},\mathcal{A}}(\lambda, L, N)] \leq 1/2 + \epsilon(\lambda)$

5 ASC from Zero-Knowledge Arguments

In this section, we show two constructions of ASC using two different zero-knowledge arguments: one is limited to a (trusted) Structured Reference String (SRS) setup, while the second can be from any Common Reference String (CRS) setup. We refer to these two ASC constructions as SRS-ASC and CRS-ASC, respectively. An SRS setup creates a reference string with a special structure (see section 3.1.3) which enables efficient ZKAs for special computations such as algebraic hash functions.

5.1 SRS-ASC

Our first construction leverages the special structure of the SRS to efficiently verify the knowledge of inputs to algebraic hash digests while ensuring that some or all inputs remain confidential. For this construction, we rely on an algebraic hash function:

$$\mathsf{Hash}^n_{srs} : (\mathbb{K}, [0,1]^*) \to [0,1]^n, \tag{2}$$

which requires a structured reference string (srs). This function takes a random secret key from the set $\mathbb{K}$ and any input, referred to as $tag \in [0,1]^*$, to produce a hash digest when $n \geq 2\lambda$. Different algebraic circuits can be used to prove that the digest was created correctly without revealing the secret key. For instance, some zero-knowledge proof protocols like anonymous messaging [85] allow to confirm that such a hash was generated using a blinding key committed in one of the commitments, without revealing which commitment has the key. To prove the security of our construction, we rely on collision-resistance of the hash function Hash^n_{srs}.

We introduce our first anonymous self-credential system $\mathsf{ASC}_{\mathsf{SRS}}$ using such a hash function. Let there be a zero-knowledge relation $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ enforced by $(\mathtt{Setup}_{\mathsf{ASC}_{\mathsf{SRS}}}, \mathcal{P}_{\mathsf{ASC}_{\mathsf{SRS}}}, \mathcal{V}_{\mathsf{ASC}_{\mathsf{SRS}}})$ for the following relation:

$$\begin{aligned} &\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}} : (srs, u := (\Lambda, tag, nul, \pi(\phi)), w := (j, k_j)) \\ &\Leftrightarrow \begin{pmatrix} C_j = \mathsf{Com}^{\mathbb{K}}_{srs}(k_j \in \mathbb{K}) \in \Lambda \wedge \\ nul = \mathsf{Hash}^n_{srs}(k_j, tag) \wedge j \in J_{WEE} \wedge |J_{WEE}| \geq 2 \end{pmatrix}. \end{aligned} \tag{3}$$

In this protocol, each prover j generates the master credential from a commitment and its blinding key, and the anonymity set Λ is formed from the commitments $[C_j]^N_{j=1}$ of all provers. The prover who wants to register a pseudonym ϕ generates a hash $nul = \mathsf{Hash}^n_{srs}(k_j, v_l)$, where k_j is the blinding key associated with her commitment, and v_l is the verifier name. The proof guarantees that nul was created correctly using the blinding key from one of the commitments without revealing any information about the commitment index j or the blinding key itself. We can denote this zero-knowledge relation in the multi-prover setting, where the adversary may corrupt provers other than the honest provers in J_{WEE} during the WEE of Definition 4. In such case, an honest prover's index j can be kept within J as long as $|J_{WEE}| \geq 2$. Also, note that we define it to be strongly unforgeable for an initial challenge ϕ as stated in Definition 6.

We build ASC from the above-mentioned zero-knowledge argument relation as follows:

▷ Here, the verifiers can select their identifiers $[v_l]^L_{l=0}$ or can be assigned identifiers as long as they are unique.
$\mathsf{ASC}_{\mathsf{SRS}}.\mathsf{Setup}(\lambda, L)$:
  return $crs := (srs = \mathtt{Setup}_{\mathsf{ASC}_{\mathsf{SRS}}}(\lambda), [v_l]^L_{l=0})$
$\mathsf{ASC}_{\mathsf{SRS}}.\mathsf{Gen}(crs, sk \in \mathbb{K})$:
  return $\Phi := \mathsf{Com}^{\mathbb{K}}_{crs}(sk)$
$\mathsf{ASC}_{\mathsf{SRS}}.\mathsf{Prove}(crs, \Lambda, l, j, sk_j, \phi)$: ▷ recall that $v_l \in crs$
  $nul := \mathsf{Hash}^n_{srs}(sk_j, v_l)$
  $\pi := \mathcal{P}_{\mathsf{ASC}_{\mathsf{SRS}}}(u(\phi) = (\Lambda, v_l, nul), w = (j, sk_j))$
  return (nul, π)
$\mathsf{ASC}_{\mathsf{SRS}}.\mathsf{Open}(crs, l, nul, \Phi, sk_j)$:
  return $nul = \mathsf{Hash}^n_{srs}(sk_j, v_l)$
$\mathsf{ASC}_{\mathsf{SRS}}.\mathsf{Verify}(crs, \Lambda, l, \phi, nul, \pi)$: ▷ recall that $v_l \in crs$
  return $\mathcal{V}_{\mathsf{ASC}_{\mathsf{SRS}}}(u(\phi) = (\Lambda, v_l, nul, \pi))$

We show that $\mathsf{ASC}_{\mathsf{SRS}}$ is a secure a anonymous self-credential protocol with Theorem 1 as follows:

Theorem 1. $\mathsf{ASC}_{\mathsf{SRS}}$ *provides correctness, robustness, Sybil resistance, unforgeability, anonymity, and multi-verifier unlinkability when the relation* $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ *is obtained by* $(\mathtt{Setup}_{\mathsf{ASC}_{\mathsf{SRS}}}, \mathcal{P}_{\mathsf{ASC}_{\mathsf{SRS}}}, \mathcal{V}_{\mathsf{ASC}_{\mathsf{SRS}}})$ *with unforgeability (Definition 6) for a commitment scheme* Com *that is hiding and binding (see proofs in Section 5.1.1).*

5.1.1 Security Proofs of SRS-ASC

Correctness In the $\mathsf{ASC}_{\mathsf{SRS}}$ protocol, the nullifiers are derived from the master secret key sk_j and the verifier identifier $v_l \in crs$. Specifically, the nullifier is calculated as $nul = \mathsf{Hash}^n_{srs}(sk_j, v_l)$. As a result, the nullifier remains consistent for any number of proofs submitted for the same verifier when the same master credential is used. Conversely, due to collision resistance of the underlying hash function, if any component differs, the nullifier will change, allowing honest provers to register with other verifiers. Therefore, given the ZKA is complete, we conclude that $\mathsf{ASC}_{\mathsf{SRS}}$ is correct as outlined in Definition 7.

Robustness Robustness, as described in Definition 8, indicates that honest provers should be able to generate proofs, even if the anonymity set includes master identities that resemble honest identities but are not correctly generated. We demonstrate the robustness of $\mathsf{ASC}_{\mathsf{SRS}}$ by leveraging the completeness of the zero-knowledge relation $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ (as shown in Eq. (4)). According to the completeness of ZKA outlined in Definition 4, as long as the statement is valid, a legitimate proof can be produced. Therefore, since the statement of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ only considers the correctness of the honest provers' commitment creation and is not affected by other commitments in the anonymity set, robustness is upheld.

Unforgeability Unforgeability of ASC implies that an adversary who does not possess the master secret key cannot produce a valid proof for a new pseudonym. Assuming that we obtain strong unforgeability for $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ and the commitments are binding (see Definition 3), $\mathsf{ASC}_{\mathsf{SRS}}$ achieves unforgeability, as defined in Definition 6 since we replaced the initial message of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ with the pseudonym.

Sybil-Resistance We demonstrate that $\mathsf{ASC}_{\mathsf{SRS}}$ is Sybil-resistant, as defined in Definition 10. This is due to the knowledge soundness of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ and the binding property of the commitment scheme (see Definition 3). Specifically, a proof can only be generated if the nullifier satisfies the condition $nul = \mathsf{Hash}^n_{srs}(sk_j, v_l)$, where sk_j is the master secret key and v_l is the verifier identifier. As a result, the adversary cannot generate two different nullifiers from the same master secret key and verifier identifier. Additionally, the adversary is unable to find two different blinding keys (master secret keys) for the same commitment, which represents the master identity. Therefore, we conclude that $\mathsf{ASC}_{\mathsf{SRS}}$ provides Sybil-resistantace.

Anonymity Anonymity in ASC ensures that, given a nullifier and a proof, no polynomial-time adversary can determine the master identity that generated them—even if the adversary selected the pseudonym. $\mathsf{ASC}_{\mathsf{SRS}}$ inherits this anonymity from the Perfect Special Honest-Verifier Zero-Knowledge (SHVZK) property of the ZKA for the relation $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$. This property ensures that both the prover's index and the master secret key remain hidden within the witness. Consequently, the adversary gains no information about which master credential was used to generate the proof, preserving anonymity.

Multi-Verifier Unlinkability Multi-verifier unlinkability entails that, when presented with two (proof, nullifier) pairs, an adversary cannot determine whether they originated from the same credential or from different credentials.

The relation $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ possesses the WEE property, which ensures that the proofs generated by $\mathsf{ASC}_{\mathsf{SRS}}$ do not disclose the index of the master identity. Furthermore, the hash function Hash^n_{srs}

is designed to be one-way and is based on a master secret key that has high entropy and uniform randomness. As a result, the nullifiers produced are pseudorandom strings. Therefore, we conclude that no p.p.t. adversary can ascertain whether the two pairs of proofs and nullifiers were generated from the same master credential or from two distinct master credentials. This establishes that $\mathsf{ASC}_{\mathsf{SRS}}$ provides multi-verifier unlinkability assuming that the ZKA of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ holds.

5.2 CRS-ASC

In the previous construction, a special setup for the hash function was required to maintain the secret key. This special setup necessitated a certain level of trust [85], which may not be ideal for global identity management. As an alternative, we present a different construction that can be built using classic CRSs.

In this new construction, we utilize multi-value commitments as master identities. Each master identity is embedded with randomly selected nullifiers for each verifier during the generation process. During registration, the prover reveals the corresponding nullifier for the verifier, proving it is committed within the overall commitment without disclosing any other nullifier. We rely on the binding property to ensure the consistency of the nullifier, meaning the prover cannot open a different nullifier and register multiple times for the same verifier using the same master identity.

Let there be a tuple of $(\mathtt{Setup}_{\mathsf{ASC}_{\mathsf{CRS}}}, \mathcal{P}_{\mathsf{ASC}_{\mathsf{CRS}}}, \mathcal{V}_{\mathsf{ASC}_{\mathsf{CRS}}})$ that ensure the following zero-knowledge relation:

$$\begin{aligned}\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}} : \left(crs, u := \left(\Lambda, l, nul, \pi(\phi)\right), w := \left(j, k_j, [nul_i]_{i=1}^{L}\right)\right) \\ \Leftrightarrow \begin{pmatrix} C_j = \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}([nul_i]_{i=1}^{L}, k_j \in \mathbb{S}) \wedge (1 \leq l \leq L) \\ \wedge (nul_l = nul) \wedge (j \in J_{WEE}) \wedge |J_{WEE}| \geq 2 \end{pmatrix}.\end{aligned} \tag{4}$$

We build an ASC scheme from the above-mentioned zero-knowledge argument relation as follows:

▷ Each verifier is assigned an index l where $l \in [1, L]$. The index l can be assigned based on the sorted verifier identifiers.
$\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Setup}(\lambda, L)$:
  return $crs := (crs = \mathtt{Setup}_{\mathsf{ASC}_{\mathsf{CRS}}}(\lambda))$ ▷ This crs defines the master secret key space to be $\mathbb{K} = \mathbb{Z}_q^L \times \mathbb{S}$.
$\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Gen}(crs, sk = ([nul_i]_{i=1}^{L} \in \mathbb{Z}_q^L, k_j \in \mathbb{S}))$:
  return $\Phi := \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}([nul_i]_{i=1}^{L}, k_j)$
$\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Prove}(crs, \Lambda, l, j, sk_j = (k_j, [nul_i]_{i=1}^{L}), \phi)$:
  $\pi := \mathcal{P}_{\mathsf{ASC}_{\mathsf{CRS}}}(u(\phi) = (\Lambda, l, nul), w = (j, k_j, [nul_l]_{l=1}^{L}))$
  return (nul, π)
$\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Open}(crs, l, nul, \Phi, sk_j = (k_j, [nul_i]_{i=1}^{L}))$:
  return $\Phi = \mathsf{Com}_{crs}^{\mathbb{Z}_q,\mathbb{S}}([nul_i]_{i=1}^{L}, k_j)$
$\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Verify}(crs, \Lambda, l, \phi, nul, \pi)$:
  return $\mathcal{V}_{\mathsf{ASC}_{\mathsf{CRS}}}(u(\phi) = (\Lambda, l, nul, \pi))$

We show that $\mathsf{ASC}_{\mathsf{CRS}}$ is a secure anonymous self-credential protocol as stated in the following theorem:

Theorem 2. $\mathsf{ASC}_{\mathsf{CRS}}$ *provides correctness, robustness, Sybil resistance, unforgeability, anonymity, and multi-verifier unlinkability when the system of* $(\mathtt{Setup}_{\mathsf{ASC}_{\mathsf{CRS}}}, \mathcal{P}_{\mathsf{ASC}_{\mathsf{CRS}}}, \mathcal{V}_{\mathsf{ASC}_{\mathsf{CRS}}})$ *satisfies the ZKA relation* $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ *with unforgeability (Definition 6) and* Com *is hiding and binding (see proofs in Section 5.2.1).*

5.2.1 Security Proofs of CRS-ASC

Correctness In $\mathsf{ASC}_{\mathsf{CRS}}$, each nullifier nul_l for verifier l are committed to the master public key Φ. Hence, the prover has to reveal the same nullifier to a corresponding verifier for any number of proofs.

Additionally, for two different verifiers: verifier l and verifier $l'(l \neq l')$, the prover will open different nullifiers if the prover committed two different nullifiers $nul_l \neq nul_{l'}$ during the generation of Φ. Assuming that all variables in the master secret key $sk = ([nul_i]_{i=0}^{l}, k)$ are chosen at random from $\mathbb{K} = \mathbb{Z}_q^L \times \mathbb{S}$, (1) the provers can register with different verifiers and (2) multiple provers can register with the same verifier. Therefore, we conclude that $\mathsf{ASC}_{\mathsf{CRS}}$ is correct.

Robustness Robustness of ASC means that an honest prover can generate a proof even when the anonymity set includes master identities that resemble correct ones but that were not correctly generated from $\mathsf{ASC}_{\mathsf{CRS}}.\mathsf{Prove}()$. We claim that $\mathsf{ASC}_{\mathsf{CRS}}$ is robust because the completeness of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ guarantees that the honest prover can create a proof as long as the statement is valid, regardless of the presence of other looks-like master identities within the anonymity set.

Strong Unforgeability Strong unforgeability means that a proof for a fresh pseudonym cannot be created even if the adversary has access to previous proofs and pseudonyms. $\mathsf{ASC}_{\mathsf{CRS}}$ obtains strong unforgeability due to the strong unforgeability of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ since the proofs are bound to the pseudonym as explained in Definition 6.

Sybil Resistance When utilizing commitments as master identities, the prover must reveal what was initially committed due to the binding property of the commitments (see Definition 3). Consequently, the prover is required to open the same nullifier l to the verifier, regardless of the pseudonym used. This process ensures the Sybil resistance expected in Definition 10 as long as commitments are binding.

Anonymity The anonymity property of the $\mathsf{ASC}_{\mathsf{CRS}}$ construction follows from the WEE property of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ since the proofs do not reveal anything about an honest prover's index of the master identity as long as there exists one more honest prover such that $|J_{WEE}| \geq 2$.

Multi-Verifier Unlinkability The WEE of $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ ensures that the proofs and the revealed nullifier do not disclose any information about the other committed nullifiers embedded with the master identity. As a result, given two pairs of proofs and nullifiers, a p.p.t. adversary cannot determine whether they were generated from the same master credential or from two different credentials. Therefore, we conclude that $\mathsf{ASC}_{\mathsf{CRS}}$ provides multi-verifier unlinkability when the commitments are hiding, as outlined in Definition 2, and $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ satisfies the WEE as in Definition 4.

6 The U2SSO System

U2SSO poses as an alternative to traditional federated identity management solutions, which allow users to access multiple service providers (SPs) using and managing just one set of credentials provided by an IdP, leveraging SSO. The proposed system, starting from the concept of self-sovereign identities (SSI), aims to deconstruct the well-established SSO user authentication flow, introducing a new composition that substitutes the centralized trusted entity with an immutable identity registry, as already happens in SSI, but provides additionally:

- a mechanism to self-derive valid *service-specific identities* linked to a single publicly available SSI,
- a primitive to prove the ownership of such SSI hidden in an *anonymity set*, and
- a novel workflow where interactions with the IdR only occur for SSI registration, eliminating timing attacks.

6.1 Model

The U2SSO system encompasses three entities.

- *User ($\mathcal{U}$)* – A user is interested in creating a digital identity to access online services. Ideally, she has complete control over the identity; she can manage just one set of master credentials while maintaining privacy across multiple services.
- *Service Provider (SP)* – A service provider is an entity advertising a service. Typically, it wants to univocally identify users both to maintain long-lived accounts for operational purposes and to protect the service against Sybil attacks for security purposes. Therefore, a user must prove ownership of a master identity on the IdR to register a pseudonym as her username for a long-lived account, and she should be limited to generating just one pseudonym from the same master identity for a specific service. Each SP may prefer a specific format for the pseudonym (i.e., email address) thus we assume it uses a preferred credential generator $\mathcal{G}_{auth}$ that conforms to a generic interface (see Section 6.2). Additionally, it is crucial for a service provider to be uniquely identifiable by users to prevent man-in-the-middle attacks. Each service provider is assigned a unique identifier known as a *service name* and an *origin*. This helps users verify whether the connection is secure and the service is legitimate. Further details on SP registration are out of the scope of this work.
- *Identity Registry (IdR)* – The identity registry acts as a semi-trusted intermediary entity that publicly stores the users' identities, maintaining one or more anonymity sets, and the service names and origin of legitimate SPs. This entity is not entrusted with credential issuing or keeping secrets, and it may actively collude with SPs to deanonymize users. However, it must guarantee that every correctly registered identity can be retrieved and read consistently. Ideally, the IdR has a Sybil-resistant onboarding process that allows users to register a master identity. This process can vary in complexity; it may involve a monetary fee for general-purpose services or require more rigorous methods, subject to real-world identification (e.g., driving licenses) for sensitive services.

6.2 Cryptographic tools

The U2SSO system employs three main cryptographic tools. First, it employs an ASC protocol (Section 4) to prove ownership of a single master identity hidden within a publicly available anonymity set while generating a nullifier for one-time identification with an SP. Second, it uses a hash-based key derivation function $\mathsf{HKDF}^{n}_{m,m'}$ (Section 3.1.1) to deterministically derive child credentials from a single master credential. Finally, it incorporates a credential generator $\mathcal{G}_{auth}$, which abstracts the details of most common authentication mechanisms and provides the following generic interface:

1: $\mathcal{G}_{auth}.\mathsf{Setup}(\lambda) : crs \triangleright$ public parameters' generation
2: $\mathcal{G}_{auth}.\mathsf{Gen}(sk \in [0,1]^n) : \phi \triangleright$ pseudonym generation
3: $\mathcal{G}_{auth}.\mathsf{Prove}(sk, W) : \sigma \triangleright$ proof against a random challenge W
4: $\mathcal{G}_{auth}.\mathsf{Verify}(\phi, W, \sigma) : 1/0 \triangleright$ proof verification

6.3 Workflow

The U2SSO system can be specified through a setup phase, and its four protocols: master identity registration, pseudonym derivation and registration, and authentication to a service, pseudonym update or revocation. We detail each and discuss their system goals.

6.3.1 Setup

The setup requires every entity to initialize its state with the necessary data to bootstrap the system. The IdR allows L service providers to register to the U2SSO system. For each SP_l, for $l \in [1, L]$, the IdR records the service name v_l, the origin, and the preferred credential generator $\mathcal{G}^{l}_{auth}$. The IdR ensures

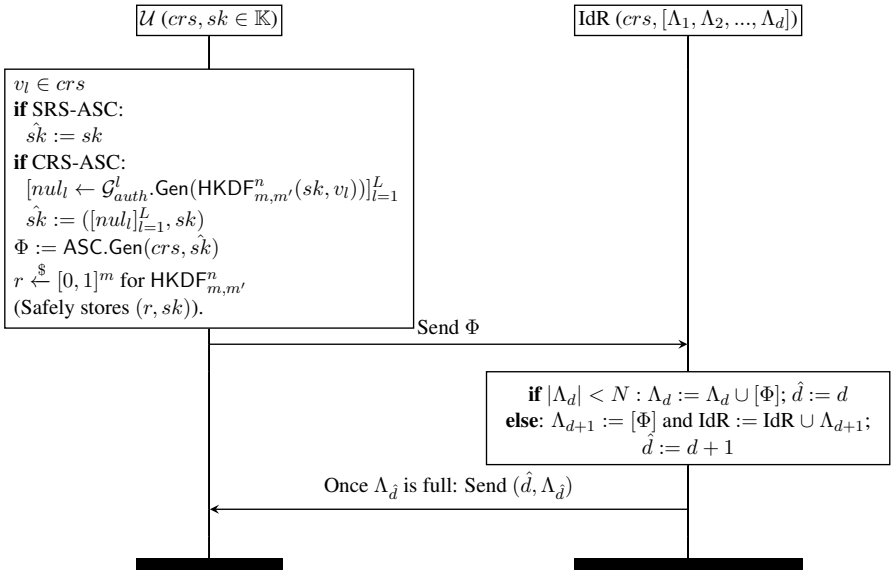


Figure 1. U2SSO master credential creation and master identity registration with the IdR

that all service names are unique, and rejects any duplicates. Service names are used in ASC as verifier identifiers, which we recall are part of the ASC public parameters.

6.3.2 Master identity registration

The protocol takes place between a user $\mathcal{U}$ and the IdR, as shown in Figure 1. After setup, $\mathcal{U}$ starts generating U2SSO credentials. Depending on which construction it is used, the generation process will be different. For SRS-ASC, $\mathcal{U}$ randomly picks sk from the key space ($sk \xleftarrow{\$} \mathbb{K}$), and from it generates the master identity Φ. However, for CRS-ASC, she first picks a key sk from the set $\mathbb{S}$ and generates nullifiers as follows:

$$[nul_l \leftarrow \mathcal{G}^l_{auth}.\mathsf{Gen}(\mathsf{HKDF}^n_{m,m'}(sk, v_l))]^L_{l=1}.$$

where $\mathcal{G}^l_{auth}$ is SP_l's chosen credential generator. Each nullifier can be deterministically recomputed and serves as a public key, whose associated secret is derived using a key derivation function with key sk and verifier name v_l as the salt. The master secret key for CRS-ASC is the pair $([nul_l]^L_{l=1}, sk)$, from which the master identity Φ is generated.

Finally, she picks a random input key material r for $\mathsf{HKDF}^n_{m,m'}$, required for future pseudonym registration. We refer to the tuple (Φ, sk, r) as U2SSO master credential, where Φ is the master identity (public), and (sk, r) is the secret key.

In U2SSO, we configure the IdR to store master identities using multiple anonymity sets, denoted by $\Lambda_1, \Lambda_2, \ldots, \Lambda_d$. The number of anonymity sets d expands as new identities are added. Each anonymity set has cardinality N, with $N \gg 2$.

Once the master credential is generated, $\mathcal{U}$ sends the master identity Φ to the IdR. The IdR fills the anonymity sets on a first-come, first-served basis, i.e., the IdR places Φ in the most recent unfilled anonymity set Λ_d or creates a new anonymity set Λ_{d+1} to store the master identity. Once an anonymity set $\Lambda_{\hat{d}}$ is full (N identities are stored), its master identities are considered *ready to use*. Only then does the IdR send that anonymity set $\Lambda_{\hat{d}}$ along with the index $\hat{d}$ back to $\mathcal{U}$, successfully completing registration.

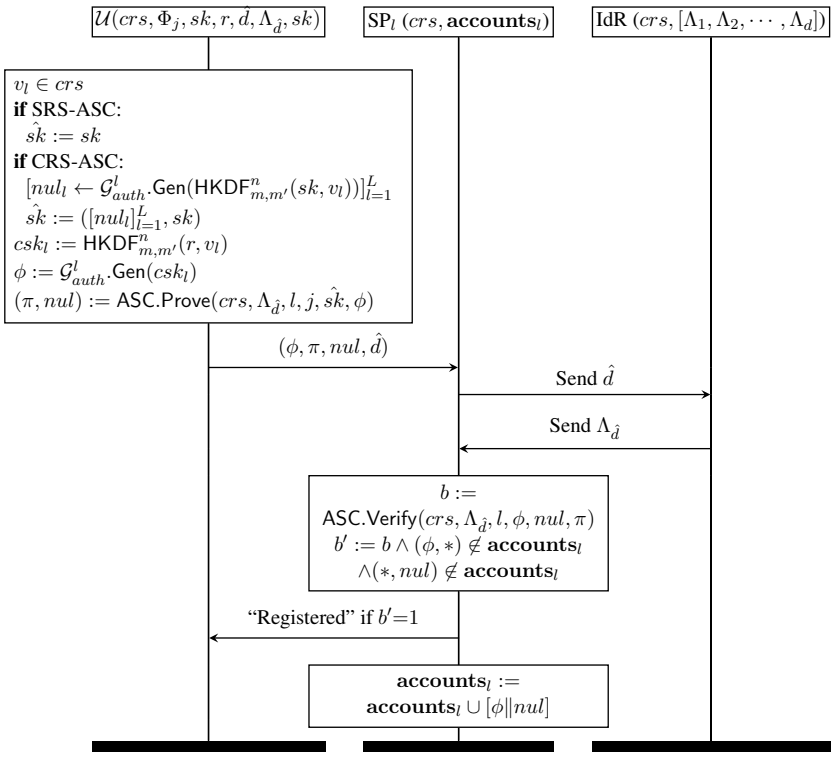


Figure 2. Registration of a service-specific pseudonym ϕ with service provider SP$_l$.

6.3.3 Pseudonym generation and registration

The protocol involves a user $\mathcal{U}$, the IdR, and a specific service provider SP_l, as shown in Figure 2. To provide an authentication flow adjacent to classic SSO, we want $\mathcal{U}$ to manage only the U2SSO master credentials and be able to derive deterministically service-specific identities as child credentials to register with different SPs. Let's recall that $\mathcal{U}$ stores (Φ_j, sk, r), where here Φ_j indicates $\mathcal{U}$ owns the jth master identity in a certain anonymity set.

Therefore, to register with a service provider SP_l, $\mathcal{U}$ generates a child secret key csk_l from $\mathsf{HKDF}^n_{m,m'}$ with r as the input key material and service name v_l as salt:

$$csk_l := \mathsf{HKDF}^n_{m,m'}(r, v_l) \in [0,1]^n \tag{5}$$

$\mathcal{U}$ feeds csk_l key to SP_l's chosen credential generator $\mathcal{G}^l_{auth}$ and generates the pseudonym as follows: $\phi := \mathcal{G}^l_{auth}.\mathsf{Gen}(csk_l)$.

Then, $\mathcal{U}$ computes the proof that she owns a master identity in the anonymity set $\Lambda_{\hat{d}}$ and a nullifier, and sends them to SP$_l$. Additionally, $\mathcal{U}$ sends $\hat{d}$, index of the anonymity set, and ϕ.

SP$_l$ then retrieves $\Lambda_{\hat{d}}$ from the IdR to verify the proof and the nullifier. This method of fixed-sized and precisely-indexed anonymity sets reduces interactions with the IdR, allowing both $\mathcal{U}$ and SP$_l$ to cache them for future use. Also, this prevents side-channel attacks on unlinkability through the timing of IdR interactions. SP$_l$ maintains an array, $\mathbf{accounts}_l$, tracking registered users. Thus, upon receiving $\mathcal{U}$'s request, it additionally checks that both ϕ and nul were not previously received to successfully conclude registration.

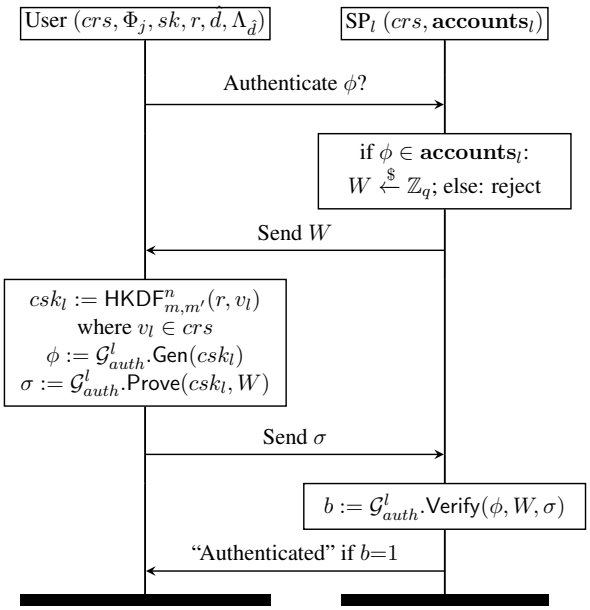


Figure 3. A proof-of-concept authentication with a registered service-specific pseudonym ϕ when $\phi \in \mathbf{accounts}_i$.

6.3.4 Authentication to a service

The protocol runs only between the user $\mathcal{U}$ and a specific service provider SP$_l$. We use a generic authentication protocol based on the credential generator interface introduced in Section 6.2. Once the pseudonym ϕ is registered to SP$_l$, $\mathcal{U}$ can log in by authenticating herself, as shown in Figure 3. First, the user requests to authenticate herself with ϕ. Then, SP$_l$ picks a random challenge W and sends it to the user if $\phi \in \mathbf{accounts}_l$. The user then proves the knowledge of the secret csk_l against the challenge W by creating a proof σ. If the proof is valid, the verifier allows $\mathcal{U}$ to access the account under ϕ.

6.3.5 Pseudonym update or revocation

A user $\mathcal{U}$ should be able to update a pseudonym used for authentication with an SP$_l$ in the event of credential updates or revocations, even if the secret key csk_l may have been compromised. U2SSO outlines two protocols designed for this situation based on the underlying ASC construction.

$\mathcal{U}$ wants to revoke a registered pseudonym ϕ with the verifier l. First, $\mathcal{U}$ notifies the verifier that she intends to authenticate herself for the registered ϕ using the knowledge of (sk, r), rather than with the potentially compromised key csk. In response, the verifier sends a random challenge $W \xleftarrow{\$} \mathbb{Z}_q$. Upon receiving W, $\mathcal{U}$ proves her identity to the verifier.

For SRS-U2SSO, she creates an ASC proof on $W \| \phi$,

$$(\pi, nul') := \mathsf{ASC.Prove}(crs, \Lambda_{\hat{d}}, l, j, \hat{sk} = sk, W \| \phi)$$

The verifier checks if the nullifier nul used to register ϕ is equal to nul', and then revokes the access to ϕ. Here, $\mathcal{U}$ effectively proves the ownership of ϕ via the knowledge of sk, without using leaked csk_l. Moreover, due to the randomness of W, the verifier ensures that adversaries can not *replay* old ASC proofs. Similarly, $\mathcal{U}$ can update ϕ to a non-empty ϕ' by generating the proof on $W \| \phi \| \phi'$.

In CRS-U2SSO, $\mathcal{U}$ shows just the knowledge of the nullifier's secret key by creating σ as follows:

$$\sigma := \mathcal{G}^l_{auth}.\mathsf{Prove}(nul_l, W \| \phi) \text{ where } nul_l := \mathsf{HKDF}^n_{m,m'}(sk, v_l).$$

As before, $\mathcal{U}$ can update ϕ to a new ϕ' by generating σ for $W \| \phi \| \phi'$.

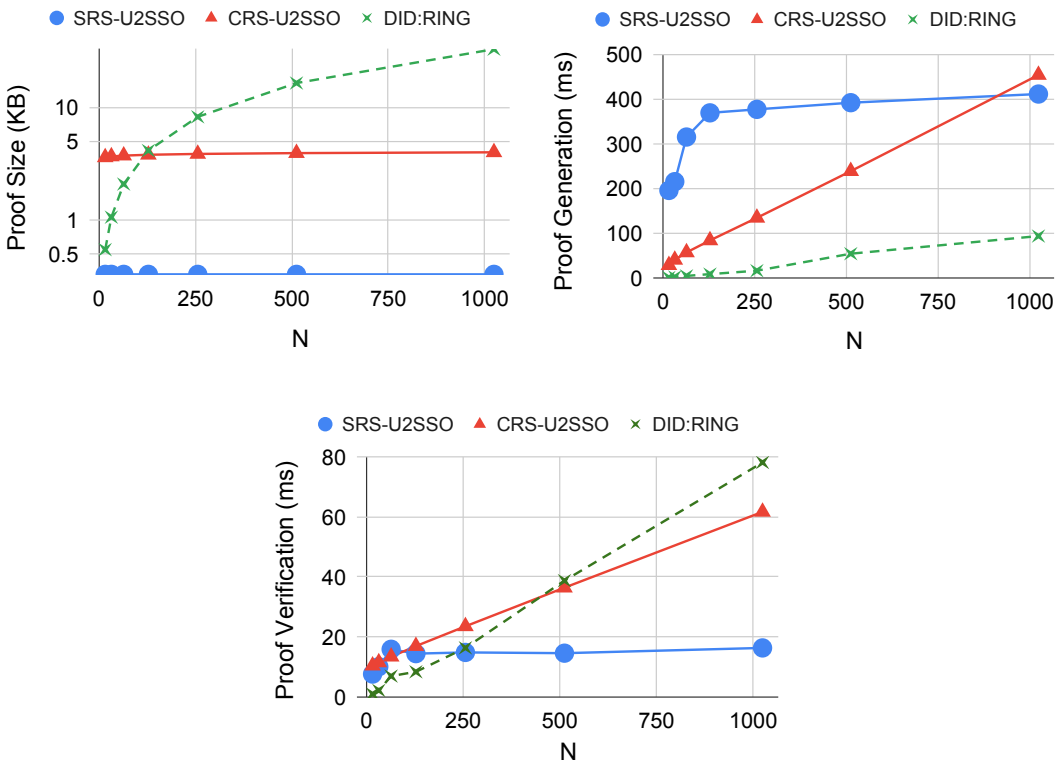


Figure 4. Proof sizes, proof generation times, and proof verification times for service registrations. Here, we compare our U2SSO constructions along with DID:RING proofs [61]. Note: N is the cardinality of Λ.

7 Implementation and Evaluation

We implement a proof-of-concept U2SSO system providing (1) open-source cryptographic libraries for CRS-U2SSO and SRS-U2SSO, implementing respectively the ASC-CRS and ASC-SRS constructions; (2) an Ethereum smart contract as the IdR that holds the anonymity sets, and (3) a user-side command line interface (CLI) and (4) a website prototype, both enabling registration and authentication with U2SSO (`https://github.com/BoquilaID/U2SSO`).

We then benchmark the relevant U2SSO cryptographic primitives and test the smart contract on Truffle [89] to estimate Gas costs for functionalities in a controlled environment.

7.1 Implementation details

CRS-U2SSO implements the zero-knowledge relation $\mathcal{R}_{\mathsf{ASC}_{\mathsf{CRS}},N,J_{WEE}}$ utilizing SHA256 and the Double-Blind Proof of Existence (DBPoE) [2] designed for multi-generator Pedersen commitments. In our implementation, we replaced the DBPoE membership proofs with a more efficient one as described in Bootle et al. [19], building it from scratch, using SECP256K1 elliptic curve [14]. In this context, a multi-value commitment consists of 33 bytes, while a field element $\mathbb{Z}_q$ consists of 32 bytes.

SRS-U2SSO, instead, instantiates $\mathcal{R}_{\mathsf{ASC}_{\mathsf{SRS}},N,J_{WEE}}$ via Semaphore/Circom (JavaScript/Rust) [31, 85] on the BN254 curve (a commitment is 96 bytes and $\mathbb{Z}_q$ elements are 32 bytes), and uses Poseidon3 hash function [48] for Hash^n_{srs}.

As a credential generator and authentication mechanism for services, SRS-U2SSO uses BLS signatures [18], while CRS-U2SSO employs Schnorr signatures [84].

The Solidity smart contract on the Ethereum blockchain suitably implements the U2SSO identity registry. The contract requires a transaction fee for U2SSO master credential registration, discouraging users from creating multiple identities due to the financial cost and Ethereum's inherent limitations on maintaining unlinkable accounts. Notably, a smart contract aligns well with the semi-trusted adversarial

model of the IdR: since every transaction is publicly traceable, anyone could potentially abuse transmitted information.

The CLI application enables users to create and store U2SSO master identities on the Ethereum-based IdR, generate proofs for service registration, and compute signatures.

Finally, the website prototype accepts pseudonym registration requests based on master identities on the IdR smart contract. It can verify proof of ownership of a master credential in a specific anonymity set and signatures for authentication. Additional details on the prototype are provided in App. A.

7.2 Experimental results

We conduct our experiments on a personal laptop with Ubuntu 22.04.4 LTS and a 12th Generation Intel Core i7-1260P with 16 cores and 32GiB of RAM, and focus on the result of the following experiments: (E1) Generation and size cost of master identities (Φ) and child identities (ϕ), (E2) Prove/verification cost of the ASC proof π based on the anonymity set cardinality N, (E3) Dependency of CRS-U2SSO on the parameter L, registered SPs, and (E4) Gas cost for the interactions with the Ethereum contract.

E1. We provide in Table 2 the sizes of Φ, and ϕ, together with the computational cost of generation. The size of Φ is relevant both for the user, who needs to store it, and the IdR that stores N of them. The user should also securely store the secret material (sk, r) (64 bytes). It is worth noting that the generation of the master identity in CRS-U2SSO is the only operation parametrized by L, the number of registered SPs, while Φ size is unaffected and stays constant. Considering that registering Φ is a one-time operation, the overhead introduced by the second construction has minimal impact on the overall system.

U2SSO	Φ (Bytes)	Φ Creation (ms)	ϕ (Bytes)	ϕ Creation (ms)
SRS-U2SSO	$\mathcal{O}(1)$: 96	$\mathcal{O}(1)$: 18.9	$\mathcal{O}(1)$: 96	$\mathcal{O}(1)$: 114
CRS-U2SSO	$\mathcal{O}(1)$: 33	$\mathcal{O}(L)^*$: 97	$\mathcal{O}(1)$: 33	$\mathcal{O}(1)$: 98

Table 2. Sizes and computational cost of Φ and ϕ. (*) CRS-U2SSO assumes $L = 100$

E2. During registration for an SP, a user needs to prove ownership of a master identity. We show how the proving and verification times, as well as the proof sizes, grows with the size of the anonymity set N, ranging between 16 and 1024. Additionally, we compare our solution with the approach presented in DID:RING [61], benchmarking the Borromean ring signature [71] on Libsecp256k1 [14], which also provides a solution to prove membership of an identity hidden in anonymity set. As shown in Figure 4, SRS-U2SSO's proof size is constant, 328 bytes, while CRS-U2SSO's proof size increases logarithmically in N from 3591 to 3981 bytes. Also, we observe CRS-U2SSO exhibit a linear increase in both proof generation and verification times. In contrast, SRS-U2SSO only shows a logarithmic increase in generation and verification times as N increases. Although, comparing our U2SSO constructions with Borromean ring signatures [71] used in DID:RING, we observe that DID:RING offers faster prover time, but suffer from larger proof sizes for $N > 128$ and slower proof verification times for $N > 512$. For SSO specifically, we believe that verification time efficiency is more critical as SPs need to execute multiple verifications, even concurrently for every new user, while proof generation is an operation that the user rarely executes only during registration, modification, or revocation of a pseudonym with the SP.

E3. We show in Figure 5 how varying the size of the anonymity set impacts the prove/verification time for different values of L in CRS-U2SSO. The linear dependency from N remains clear, while changes due to the L parameter seem to only introduce a small offset. For this reason, we fix $N = 1024$ and observe in Figure 6 how Φ generation, proof size, and proving/verifying π are impacted by L. The verification time appears to be the least affected factor, which is also important for the SPs.

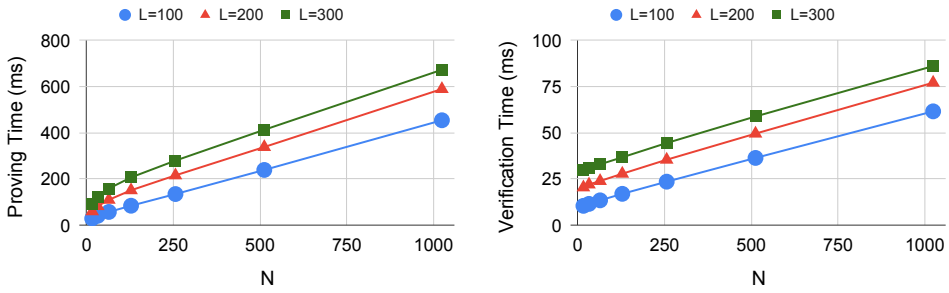


Figure 5. Proving/verifying proof π for varying N and L in CRS-U2SSO

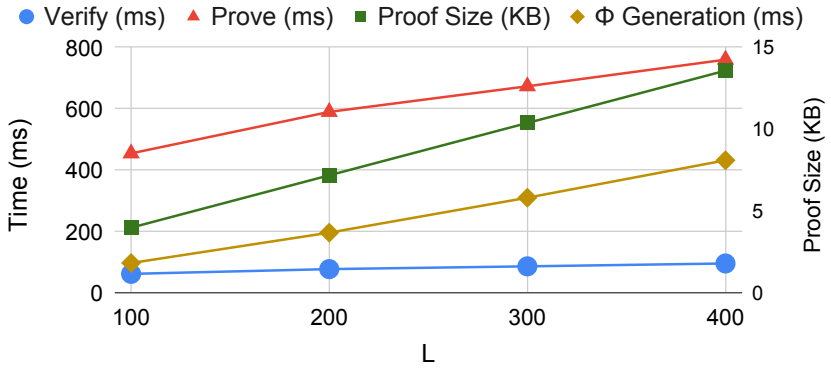


Figure 6. CRS-U2SSO with varying L and $N = 1024$

E4. We compare estimated minimal Gas costs for CRS-U2SSO solutions, DID:RING, and Hades in Table 3. The cost for storing master identities depends on the size of Φ and the gas cost, plus a registration fee that can be purposely defined in the smart contract. DID:RING register a whole set of identities to be able later to use ring signature for anonymous authentication, Hades produces a couple of ZKP, to prove knowledge of a Sybil-resistant identity, while our solution does not need to store pseudonyms or proofs on the IdR. It is worth noting that our solutions offer the best cost optimization for any pair of (N, L) due to the fixed size of Φ and the one-time write operation on the smart contract (only during registration of Φ).

Protocol	ID Storage	Registration for a service	Interaction with IdR
DID:RING	$\approx$136	$>136 + 136N$	always
Hades	339	248	always
CRS-U2SSO	136	0	one-time registration

Table 3. Cost of Ethereum-based IdR Solutions in Gwei

8 Discussion and Conclusion

We discuss U2SSO's complementary aspects, a comparison with privacy-preserving SSO solutions, and finally conclude the paper.

Verifiable Credentials (VC). A verifiable credential is an attribute about the user backed up by authenticated proof from a reputable party. U2SSO can easily add support for VC as follows: the master identity should additionally contain commitments to signed statements from an authority while anonymity sets on the IdR should be organized by grouping users possessing the same attributes, signed by the same authority, e.g., all users are over 18. An active line of research focuses specifically on how to implement VCs in a decentralized setting [58, 86, 55, 72]. However, many general-purpose services do not need VCs, and can benefit solely from the Sybil resistance and privacy of U2SSO.

Accountability. Complete unlinkability is not always desirable, as SPs must be able to blacklist misbehaving users [66]. Currently, U2SSO lacks a mechanism for revoking a master identity to ensure auditability. A common solution, used in other work [95, 92, 1], involves an external mechanism where a trusted committee can jointly decrypt a user's secret to revoke or track malicious accounts based on specific policies. However, by design, U2SSO already guarantees accountability and revocation of pseudonyms within a specific SP, which is sufficient for many Web 2.0 services.

Other privacy-preserving SSO. We group privacy-enhancing SSO solutions into two categories. The first one assumes the model with a trusted IdP and aims at mitigating *observability* of the IdP to prevent it from tracking users' activities. Some solutions [38, 4, 94] address the problem from a system design perspective introducing intermediary trusted components, others [95, 57, 52, 80, 22] from a cryptographic one. Other works [64, 45, 54, 53, 51] focus specifically on increasing the privacy of industry standards, such as OpenID Connect (OIDC) [39], by proposing solutions ready to be integrated into deployed systems. Some works have a committee acting as an IdP [69, 86] and issuing credentials. Nevertheless, specific security properties still depend on the overall trust in the committee. In all these solutions, the IdP remains a single point of failure either for service availability and timing attacks or because they require users to store multiple pseudonyms. The second category introduces SSI relying on a distributed immutable IdR as a trust anchor [30, 41, 92, 73, 16, 46]. These proposals focus mainly on obtaining unlinkable credentials for Web 3.0 services (distributed applications running on chain). We claim this is only possible if blockchain transactions are untraceable. In a different context, zkLogin [10] proposes a construction to prove ownership of credentials issued by an OAuth Provider (IdP) to authenticate blockchain transactions, converging again to the first model of the trust IdP.

Conclusion Our U2SSO solution, backed up by a novel notion, Anonymous Self-Credentials (ASC), leverages the model introduced with self-sovereign identity and utilizes a public identity registry as an immutable ledger. This approach introduces a user-centric credential system that eliminates the need for a trusted identity provider while ensuring both Sybil resistance for service providers and multi-verifier unlinkability for users. It successfully introduces a transition of Web 2.0 credentials to Web 3.0, promoting user-centric approaches in the future digital world.

Acknowledgments

We thank Ya-wen Jeng from Ethereum Foundation for introducing us to the Semaphore protocol [85] and helping with the initial construction of SRS-based solution for U2SSO implementation.

References

[1] B. Alangot, P. Szalachowski, T. T. A. Dinh, S. Meftah, J. I. Gana, K. M. M. Aung, and Z. Li, "Decentralized identity authentication with auditability and privacy," *Algorithms*, vol. 16, no. 1, p. 4, 2023.

[2] J. Alupotha, "Double-blind proof of existence for decentralized identities," *IEEE Access*, vol. 11, pp. 132180–132195, 2023.

[3] N. Amarasinghe, X. Boyen, and M. McKague, "The complex shape of anonymity in cryptocurrencies: Case studies from a systematic approach," in *Financial Cryptography (1)*, vol. 12674 of *Lecture Notes in Computer Science*, pp. 205–225, Springer, 2021.

[4] M. R. Asghar, M. Backes, and M. Simeonovski, "PRIMA: privacy-preserving identity and access management at internet-scale," in *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, pp. 1–6, IEEE, 2018.

[5] D. Atkins, W. Stallings, and P. R. Zimmermann, "PGP message exchange formats," *RFC*, vol. 1991, pp. 1–21, 1996.

[6] R. Atterer, M. Wnuk, and A. Schmidt, "Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction," in *WWW*, pp. 203–212, ACM, 2006.

[7] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in *International conference on security and cryptography for networks*, pp. 111–125, Springer, 2006.

[8] M. H. Au, W. Susilo, Y. Mu, and S. S. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Systems Journal*, vol. 7, no. 2, pp. 249–261, 2012.

[9] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, "Decentralized identity: Where did it come from and where is it going?," *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[10] F. Baldimtsi, K. K. Chalkias, Y. Ji, J. Lindstrøm, D. Maram, B. Riva, A. Roy, M. Sedaghat, and J. Wang, "zklogin: Privacy-preserving blockchain authentication with existing credentials," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 3182–3196, 2024.

[11] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings* (S. Halevi, ed.), vol. 5677 of *Lecture Notes in Computer Science*, pp. 108–125, Springer, 2009.

[12] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya, "P-signatures and noninteractive anonymous credentials," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008* (R. Canetti, ed.), vol. 4948 of *Lecture Notes in Computer Science*, pp. 356–374, Springer, 2008.

[13] P. Bichsel, C. Binding, J. Camenisch, T. Groß, T. Heydt-Benjamin, D. Sommer, and G. Zaverucha, "Cryptographic protocols of the identity mixer library. technical report RZ 3730, IBM research – Zurich," 2009.

[14] `https://github.com/bitcoin-core/secp256k1`. Optimized C library for EC operations on curve secp256k1.

[15] J. Blömer and J. Bobolz, "Delegatable attribute-based anonymous credentials from dynamically malleable signatures," in *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings* (B. Preneel and F. Vercauteren, eds.), vol. 10892 of *Lecture Notes in Computer Science*, pp. 221–239, Springer, 2018.

[16] V. Bolgouras, A. Angelogianni, I. Politis, and C. Xenakis, "Trusted and secure self-sovereign identity framework," in *ARES 2022: The 17th International Conference on Availability, Reliability and Security, Vienna,Austria, August 23 - 26, 2022*, pp. 101:1–101:6, ACM, 2022.

[17] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual international cryptology conference*, pp. 41–55, Springer, 2004.

[18] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings* (C. Boyd, ed.), vol. 2248 of *Lecture Notes in Computer Science*, pp. 514–532, Springer, 2001.

[19] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit, "Short accountable ring signatures based on DDH," in *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I* (G. Pernul, P. Y. A. Ryan, and E. R. Weippl, eds.), vol. 9326 of *Lecture Notes in Computer Science*, pp. 243–265, Springer, 2015.

[20] P. Branco and P. Mateus, "A traceable ring signature scheme based on coding theory," in *Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*, pp. 387–403, Springer, 2019.

[21] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pp. 315–334, IEEE Computer Society, 2018.

[22] J. Camenisch and E. V. Herreweghen, "Design and implementation of the *idemix* anonymous credential system," in *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002* (V. Atluri, ed.), pp. 21–30, ACM, 2002.

[23] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich, "How to win the clone wars: Efficient periodic n-times anonymous authentication," *IACR Cryptol. ePrint Arch.*, p. 454, 2006.

[24] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding* (B. Pfitzmann, ed.), vol. 2045 of *Lecture Notes in Computer Science*, pp. 93–118, Springer, 2001.

[25] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings* (M. K. Franklin, ed.), vol. 3152 of *Lecture Notes in Computer Science*, pp. 56–72, Springer, 2004.

[26] A. Chator, M. Green, and P. R. Tiwari, "Sok: Privacy-preserving signatures," *Cryptology ePrint Archive*, 2023.

[27] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of Crypto 82*, pp. 199–203, Springer, 1983.

[28] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[29] D. Chaum and E. Van Heyst, "Group signatures," in *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10*, pp. 257–265, Springer, 1991.

[30] R. Chen, F. Shu, S. Huang, L. Huang, H. Liu, J. Liu, and K. Lei, "Bidm: A blockchain-enabled cross-domain identity management system," *Journal of Communications and Information Networks*, vol. 6, no. 1, pp. 44–58, 2021.

[31] `https://docs.circom.io/background/background/`. Zero-knowledge proofs.

[32] E. Commission, "What is (european blockchain services infrastructure) ebsi?." `https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/What+is+ebsi`, 2025.

[33] E. Crites, A. Kiayias, M. Kohlweiss, and A. Sarencheh, "Syra: Sybil-resilient anonymous signatures with applications to decentralized identity," *Cryptology ePrint Archive*, 2024.

[34] I. Damgård, K. Dupont, and M. Ø. Pedersen, "Unclonable group identification," in *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings* (S. Vaudenay, ed.), vol. 4004 of *Lecture Notes in Computer Science*, pp. 555–572, Springer, 2006.

[35] J. R. Douceur, "The sybil attack," in *IPTPS*, vol. 2429 of *Lecture Notes in Computer Science*, pp. 251–260, Springer, 2002.

[36] C. Ellison *et al.*, "Establishing identity without certification authorities," in *USENIX Security Symposium*, pp. 67–76, 1996.

[37] Ethereum.org, "Welcome to ethereum." `https://ethereum.org/en/foundation/`, 2024.

[38] D. Fett, R. Küsters, and G. Schmitz, "SPRESSO: A secure, privacy-respecting single sign-on system for the web," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015* (I. Ray, N. Li, and C. Kruegel, eds.), pp. 1358–1369, ACM, 2015.

[39] D. Fett, R. Küsters, and G. Schmitz, "The web SSO standard openid connect: In-depth formal security analysis and security guidelines," in *CSF*, pp. 189–202, IEEE Computer Society, 2017.

[40] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings* (A. M. Odlyzko, ed.), vol. 263 of *Lecture Notes in Computer Science*, pp. 186–194, Springer, 1986.

[41] S. Friebe, I. Sobik, and M. Zitterbart, "Decentid: Decentralized and privacy-preserving identity storage system using smart contracts," in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 12th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2018, New York, NY, USA, August 1-3, 2018*, pp. 37–42, IEEE, 2018.

[42] E. Fujisaki, "Sub-linear size traceable ring signatures without random oracles," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 1, pp. 151–166, 2012.

[43] E. Fujisaki and K. Suzuki, "Traceable ring signature," in *International Workshop on Public Key Cryptography*, pp. 181–200, Springer, 2007.

[44] C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi, "Fiat–shamir bulletproofs are non-malleable (in the random oracle model)," *Journal of Cryptology*, vol. 38, no. 1, pp. 1–48, 2025.

[45] G. Gao, Y. Zhang, Y. Song, and S. Li, "Privsso: Practical single-sign-on authentication against subscription/access pattern leakage," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 5075–5089, 2024.

[46] L. Gao, J. Yu, J. Zhang, Y. Tang, and Q. Wen, "AASSI: A self-sovereign identity protocol with anonymity and accountability," *IEEE Access*, vol. 12, pp. 58378–58394, 2024.

[47] S. Garfinkel, *PGP: pretty good privacy*. " O'Reilly Media, Inc.", 1995.

[48] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021* (M. D. Bailey and R. Greenstadt, eds.), pp. 519–535, USENIX Association, 2021.

[49] J. Groth, "On the size of pairing-based non-interactive arguments," in *EUROCRYPT (2)*, vol. 9666 of *Lecture Notes in Computer Science*, pp. 305–326, Springer, 2016.

[50] J. Groth and Y. Ishai, "Sub-linear zero-knowledge argument for correctness of a shuffle," in *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings* (N. P. Smart, ed.), vol. 4965 of *Lecture Notes in Computer Science*, pp. 379–396, Springer, 2008.

[51] C. Guo, J. Lin, Q. Cai, F. Li, Q. Wang, J. Jing, B. Zhao, and W. Wang, "UPPRESSO: untraceable and unlinkable privacy-preserving single sign-on services," *CoRR*, vol. abs/2110.10396, 2021.

[52] H. Halpin, "NEXTLEAP: decentralizing identity with privacy for secure messaging," in *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*, pp. 92:1–92:10, ACM, 2017.

[53] S. Hammann, R. Sasse, and D. A. Basin, "Privacy-preserving openid connect," in *AsiaCCS*, pp. 277–289, ACM, 2020.

[54] J. He, L. Lei, Y. Wang, P. Wang, and J. Jing, "ARPSSO: an oidc-compatible privacy-preserving SSO scheme based on RP anonymization," in *ESORICS (2)*, vol. 14983 of *Lecture Notes in Computer Science*, pp. 268–288, Springer, 2024.

[55] J. Hesse, N. Singh, and A. Sorniotti, "How to bind anonymous credentials to humans," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023* (J. A. Calandrino and C. Troncoso, eds.), pp. 3047–3064, USENIX Association, 2023.

[56] idcommons, "Identity commons." `https://www.idcommons.org/`, 2025.

[57] M. Isaakidis, H. Halpin, and G. Danezis, "Unlimitid: Privacy-preserving federated identity management using algebraic macs," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, WPES@CCS 2016, Vienna, Austria, October 24 - 28, 2016* (E. R. Weippl, S. Katzenbeisser, and S. D. C. di Vimercati, eds.), pp. 139–142, ACM, 2016.

[58] A. D. Johnson, I. Alom, and Y. Xiao, "Rethinking single sign-on: A reliable and privacy-preserving alternative with verifiable credentials," in *Proceedings of the 10th ACM Workshop on Moving Target Defense*, MTD '23, (New York, NY, USA), p. 25–28, Association for Computing Machinery, 2023.

[59] S. A. Kakvi, K. M. Martin, C. Putman, and E. A. Quaglia, "Sok: anonymous credentials," in *International Conference on Research in Security Standardisation*, pp. 129–151, Springer, 2023.

[60] I. Karantaidou, O. Renawi, F. Baldimtsi, N. Kamarinakis, J. Katz, and J. Loss, "Blind multisignatures for anonymous tokens with decentralized issuance," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1508–1522, 2024.

[61] D. Kasimatis, S. Grierson, W. J. Buchanan, C. Eckl, P. Papadopoulos, N. Pitropakis, C. Thomson, and B. Ghaleb, "DID: RING: ring signatures using decentralised identifiers for privacy-aware identity," *CoRR*, vol. abs/2403.05271, 2024.

[62] A. Kiayias, Y. Tsiounis, and M. Yung, "Traceable signatures," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 571–589, Springer, 2004.

[63] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Annual Cryptology Conference*, pp. 631–648, Springer, 2010.

[64] M. Kroschewski, A. Lehmann, and C. Özbay, "Oppid: Single sign-on with oblivious pairwise pseudonyms," *Cryptology ePrint Archive*, 2024.

[65] E. Krul, H.-y. Paik, S. Ruj, and S. S. Kanhere, "Sok: Trusting self-sovereign identity," *arXiv preprint arXiv:2404.06729*, 2024.

[66] E. Krul, H. Paik, S. Ruj, and S. S. Kanhere, "Sok: Trusting self-sovereign identity," *CoRR*, vol. abs/2404.06729, 2024.

[67] Y. Lindell, "Parallel coin-tossing and constant-round secure two-party computation," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings* (J. Kilian, ed.), vol. 2139 of *Lecture Notes in Computer Science*, pp. 171–189, Springer, 2001.

[68] C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, and M. Sena, "Uport: A platform for self-sovereign identity," *URL: https://whitepaper. uport. me/uPort_ whitepaper_DRAFT20170221. pdf*, vol. 128, p. 214, 2017.

[69] 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 *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pp. 1348–1366, IEEE, 2021.

[70] N. B. Margolin and B. N. Levine, "Quantifying resistance to the sybil attack," in *Financial Cryptography*, vol. 5143 of *Lecture Notes in Computer Science*, pp. 1–15, Springer, 2008.

[71] G. Maxwell and A. Poelstra, "Borromean ring signatures," *Accessed: Jun*, vol. 8, p. 2019, 2015.

[72] C. Mazzocca, A. Acar, A. S. Uluagac, and R. Montanari, "EVOKE: efficient revocation of verifiable credentials in iot networks," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024* (D. Balzarotti and W. Xu, eds.), USENIX Association, 2024.

[73] O. Mir, M. Roland, and R. Mayrhofer, "Decentralized, privacy-preserving, single sign-on," *Secur. Commun. Networks*, vol. 2022, pp. 9983995:1–9983995:18, 2022.

[74] N. Naik and P. Jenkins, "uport open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain," in *IEEE International Symposium on Systems Engineering, ISSE 2020, Vienna, Austria, October 12 - November 12, 2020*, pp. 1–7, IEEE, 2020.

[75] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[76] OpenID, "How openid connect works." `https://openid.net/developers/how-connect-works/`, 2025.

[77] R. Oppliger, "Microsoft. net passport and identity management," *Information Security Technical Report*, vol. 9, no. 1, pp. 26–34, 2004.

[78] I. Ozcelik, S. Medury, J. Broaddus, and A. Skjellum, "An overview of cryptographic accumulators," *arXiv preprint arXiv:2103.04330*, 2021.

[79] E. Papadogiannakis, P. Papadopoulos, N. Kourtellis, and E. P. Markatos, "User tracking in the post-cookie era: How websites bypass GDPR consent to track users," in *WWW*, pp. 2130–2141, ACM / IW3C2, 2021.

[80] C. Paquin and G. Zaverucha, "U-prove cryptographic specification v1.1 (revision 3)," December 2013. Released under the Open Specification Promise.

[81] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings* (K. Sako, ed.), vol. 9610 of *Lecture Notes in Computer Science*, pp. 111–126, Springer, 2016.

[82] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *IACR international conference on public-key cryptography*, pp. 628–656, Springer, 2020.

[83] A. Scafuro and B. Zhang, "One-time traceable ring signatures," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26*, pp. 481–500, Springer, 2021.

[84] C. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings* (G. Brassard, ed.), vol. 435 of *Lecture Notes in Computer Science*, pp. 239–252, Springer, 1989.

[85] Semaphore, "Semaphore protocol." `https://github.com/semaphore-protocol/semaphore`, 2024.

[86] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, The Internet Society, 2019.

[87] S. Tessaro and C. Zhu, "Revisiting bbs signatures," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 691–721, Springer, 2023.

[88] X. Thanh Khuc, W. Susilo, D. H. Duong, F. Guo, K. Fukushima, and S. Kiyomoto, "Traceable ring signatures: Logarithmic-size, without any setup, from standard assumptions," in *International Conference on Provable Security*, pp. 189–208, Springer, 2024.

[89] `https://archive.trufflesuite.com/`. The most comprehensive suite of tools for smart contract development.

[90] G. Vitto and A. Biryukov, "Dynamic universal accumulator with batch update over bilinear groups," in *Cryptographers' Track at the RSA Conference*, pp. 395–426, Springer, 2022.

[91] W3C, "Identity & the Web." `https://www.w3.org`, 2025.

[92] 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 *ACSAC*, pp. 216–228, ACM, 2023.

[93] L. Wang, G. Zhang, and C. Ma, "A survey of ring signature," *Frontiers of Electrical and Electronic Engineering in China*, vol. 3, pp. 10–19, 2008.

[94] R. Xu, S. Yang, F. Zhang, and Z. Fang, "MISO: legacy-compatible privacy-preserving single sign-on using trusted execution environments," in *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*, pp. 352–372, IEEE, 2023.

[95] Z. Zhang, M. Król, A. Sonnino, L. Zhang, and E. Rivière, "EL PASSO: efficient and lightweight privacy-preserving single sign on," *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 2, pp. 70–87, 2021.

A U2SSO System with A Decentralized IdR

We explain the interactions of the U2SSO in this section. We state the Solidity contract code of a decentralized IdR for CRS-U2SSO in Listing 1. As shown in the code, this IdR only take Gas to store master identities (Φ) (`addID`). Hence, the users and the service providers can read the stored Φs from the IdR, i.e., running (`getIDs`), at no fee.

```
pragma solidity ^0.8.13;

contract U2SSO {
  struct ID {
    uint256 id;
    uint id33;
    bool active;
    address owner;
  }

  address private _owner;
  constructor() {
    _owner = msg.sender;
  }

  ID[] public idList;

  uint nextIndex;

  function addID (uint256 _id, uint _id33) public returns (uint) {
    idList.push(ID(_id, _id33, true, _owner));
    nextIndex = nextIndex + 1;
    return nextIndex - 1;
  }

  function getIDs (uint _index) public view returns (uint256, uint) {
    ID storage id = idList[_index];
    return (id.id, id.id33);
  }

  function getState (uint _index) public view returns (bool) {
    ID storage id = idList[_index];
    return id.active;
  }

  function getIDSize () public view returns (uint) {
    return nextIndex;
  }

  function getIDIndex (uint256 _id, uint _id33) public view returns (int) {
    for (uint i = 0; i < nextIndex; i++) {
      bool exists = (idList[i].id == _id) && (idList[i].id33 == _id33);
      if(exists == true) {
        return int(i);
      }
    }
    return -1;
  }
}
```

Listing 1. Decentralized IdR contract

Master identity creation The user begins by utilizing the U2SSO Command Line Interface (CLI) to create a master identity and stores it in a smart contract. This process requires the smart contract address, the output key location, and an Ethereum private key as inputs. As shown in Figure 7, the CLI outputs various details, including the current ID size or anonymity set size, which represents the total number of Φs stored in the smart contract.

Registration for Services When a user wants to sign up for a website that accepts U2SSO credentials, the website first presents the user with a challenge on the sign-up page, as shown in Figure 8a. The user then takes pseudonym ϕ for the service. With this pseudonym and the challenge, the user generates proofs membership proof π using the U2SSO CLI, as shown in Figure 8b. Here, CLI uses the concatenation of

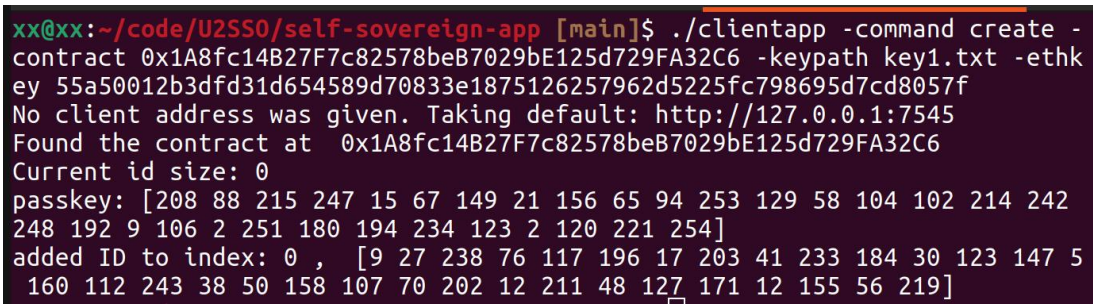
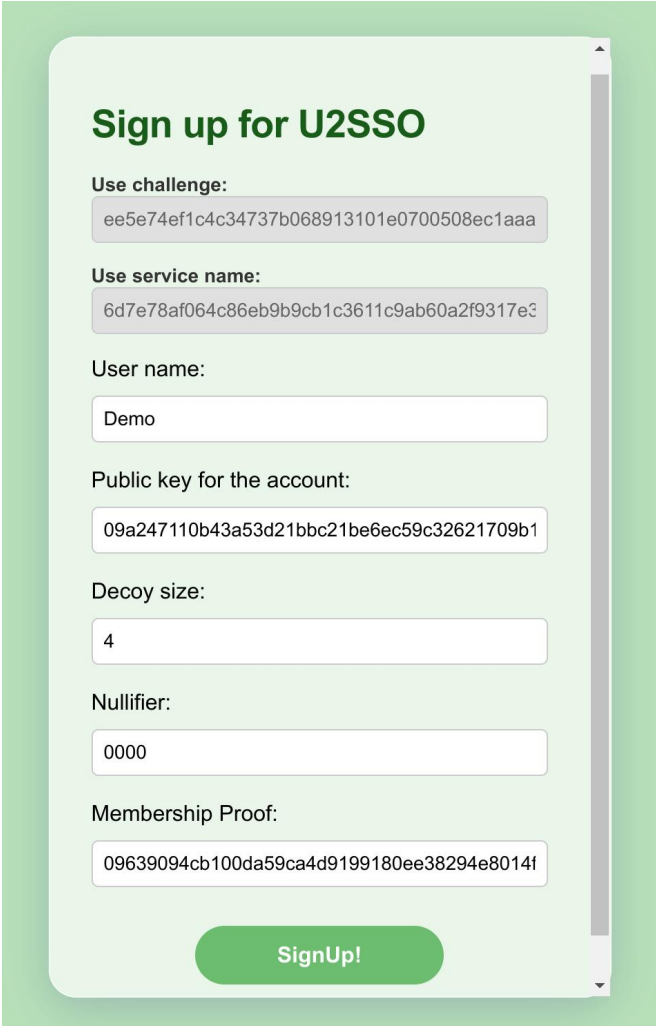


Figure 7. SSO-Id Creation

the challenge and the pseudonym as the initial message for the ASC. Subsequently, the user inserts the outputs of the CLI output into the sign-up form. The website then downloads the master identities from IdR and validates the proofs. If the proofs are valid, the website registers the user's service public key with a signup successful message.

Authentication for Services When a user wants to login to the website, the website sends the user a challenge, as shown in Figure 9a on the login page. The user then takes this challenge and generates a signature using the U2SSO CLI, as shown in Figure 9b. After that, the user inputs the signature into the login form. The website allows the user to login if the signature is valid, and a log in successful message is displayed, as shown in Figure 9c.

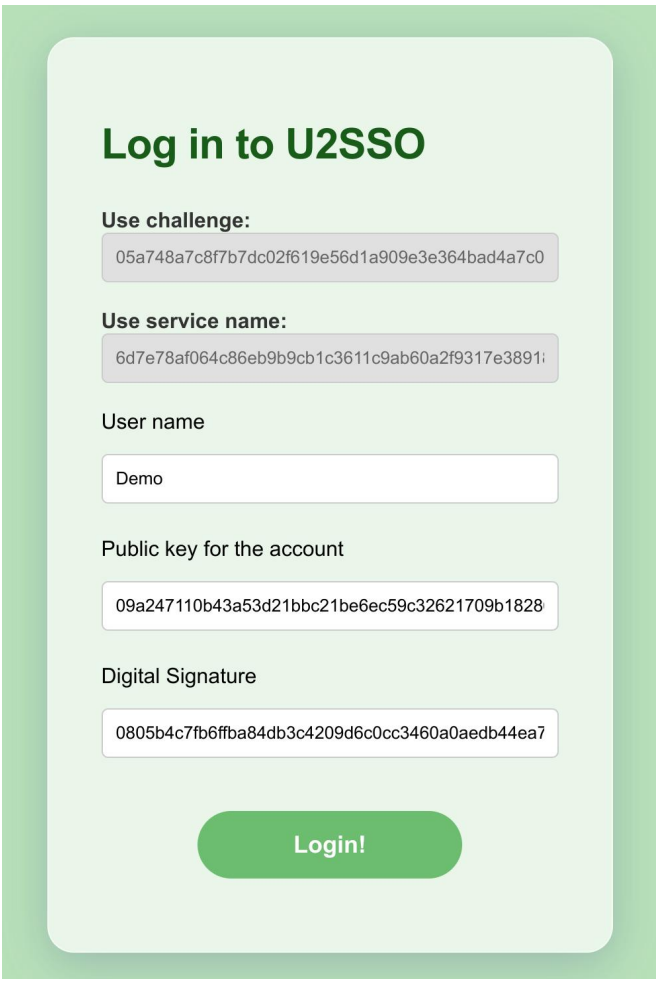


(a) SSO-Id Register Webpage

```
xx@xx:~/code/U2SSO/self-sovereign-app [main]$ ./clientapp -command register
 -contract 0x1A8fc14B27F7c82578beB7029bE125d729FA32C6 -keypath key1.txt -sn
ame 6d7e78af064c86eb9b9cb1c3611c9ab60a2f9317e3891891ef31770939f78ef8 -chall
enge ee5e74ef1c4c34737b068913101e0700508ec1aaa77b3f58dbd70309798098cf
No client address was given. Taking default: http://127.0.0.1:7545
Found the contract at  0x1A8fc14B27F7c82578beB7029bE125d729FA32C6
Current id size: 4
service name size: 64
challenge size: 64
total Id size: 4
chosen ring size: 4  and m: 2
proof hex format:  09639094cb100da59ca4d9199180ee38294e8014ff5961c8fabc7f67
f592c7ccf1091a04ddec28d8c0fb30bd357e63d8be8c85cd317306f7df7fc307e9cbceff716
9097be15db7107adfec3dd80b90eb564eefdd376c34bbb1a0a39f86110d6759c4e50908f80a
4151703f120354fb9d24fbe717ecdc7b73a5000494cb8f7e4d508242dd09a8c490ae751094f
55964929e59ff8b91c3940ba4e77e0ce6c005f943fefaca9009ce8cc78d70a1615f58fc3096
732baaf8f8eb115e73a2c9f2b51df366def633c2b308287521b61906ddd945de99ca5c1a635
b3d6475d42080dd548a6800758411a622a2dba0580f8a4ab3dbcd8b6c6017da816027321aaf
dfc167f0e2c1f9820ccdaa87628c5f1406e61790395678a5e9ca5060fe7633aadca6ab7f353
dc547757f3ae3402993b1925f260f3808e5cfcaf6a983d0f8de2f645c667ccc5fcfc6fef973
3212b71f75b0e96bedb7fa59f9a8723c9889e952473e446db555c9571f03
spkBytes hex format:  09a247110b43a53d21bbc21be6ec59c32621709b18286ff990c77
e84e06b623e72
N:  4
```

(b) SSO-Id Register CLI

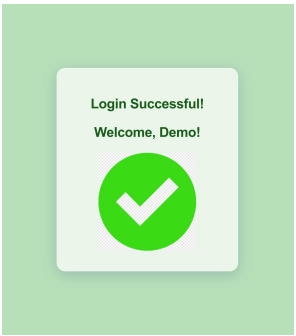
Figure 8. SSO-Id Registration Process



(a) SSO-Id Authentication Webpage

```
xx@xx:~/code/U2SSO/self-sovereign-app [main]$ ./clientapp -command auth -co
ntract 0x1A8fc14B27F7c82578beB7029bE125d729FA32C6 -keypath key1.txt -sname
6d7e78af064c86eb9b9cb1c3611c9ab60a2f9317e3891891ef31770939f78ef8 -challenge
 05a748a7c8f7b7dc02f619e56d1a909e3e364bad4a7c08ce39b164b623ab0e82
No client address was given. Taking default: http://127.0.0.1:7545
Found the contract at  0x1A8fc14B27F7c82578beB7029bE125d729FA32C6
Current id size: 4
service name size: 64
challenge size: 64
proof auth hex format:  0805b4c7fb6ffba84db3c4209d6c0cc3460a0aedb44ea7d98c7
080fc0f5731a0d9fbe58ce448bf557eb8da0698f22f16f26945d69c7ef3f755e498585462d3
975f
```

(b) SSO-Id Authentication CLI



(c) SSO-Id Authentication Successful

Figure 9. SSO-Id Authentication Process