VeriSSO: A Privacy-Preserving Legacy-Compatible Single Sign-On Protocol Using Verifiable Credentials

Ifteher Alom, Sudip Bhujel, Yang Xiao
University of Kentucky, Lexington, KY, USA
Email: {ifteheralom,sudipbhujel,xiaoy}@uky.edu

***Abstract*—Single Sign-On (SSO) is a popular authentication mechanism enabling users to access multiple web services with a single set of credentials. Despite its convenience, SSO faces outstanding privacy challenges. The Identity Provider (IdP) represents a single point of failure and can track users across different Relying Parties (RPs). Multiple colluding RPs may track users through common identity attributes. In response, anonymous credential-based SSO solutions have emerged to offer privacy-preserving authentication without revealing unnecessary user information. However, these solutions face two key challenges: supporting RP authentication without compromising user unlinkability and maintaining compatibility with the predominant Authorization Code Flow (ACF).**

This paper introduces VeriSSO, a novel SSO protocol based on verifiable credentials (VC) that supports RP authentication while preserving privacy and avoiding single points of failure. VeriSSO employs an independent authentication server committee to manage RP and user authentication, binding RP authentication with credential-based anonymous user authentication. This approach ensures user unlinkability while supporting RP authentication and allows RPs to continue using their existing verification routines with identity tokens as in the ACF workflow. VeriSSO's design also supports lawful de-anonymization, ensuring user accountability for misbehavior during anonymity. Experimental evaluations of VeriSSO demonstrate its efficiency and practicality, with authentication processes completed within 100 milliseconds.

***Index Terms*—User Authentication, Privacy-Preserving SSO, Verifiable Credentials, Anonymization, Decentralization.**

I. Introduction

Single Sign-On (SSO) is a widely used authentication method that allows users to access various web services with a single set of login credentials instead of separate logins. In a typical execution of SSO, a user attempts to log into the service portal of a Relying Party (RP) by authenticating with an Identity Provider (IdP) with whom the user has a preexisting registration. Upon authenticating the user, the IdP generates an *identity token* specifying certain user information (e.g., name, email, age) required by the RP. The IdP signs the identity token with its private key attesting to the user's identity to the RP. Currently, OpenID Connect (OIDC) [1] is the most commonly used SSO standard. OIDC is based on the popular authorization protocol OAuth 2.0 [2].

Despite SSO's convenience and wide adoption by web applications, it faces increasing privacy and security challenges. First, an IdP's central role poses a significant privacy risk to users. In OIDC's specification, the user has to inform its IdP about which RP will be accessed, enabling the IdP to track the user's accesses to different RPs. By analyzing the nature of each RP website, a curious IdP can infer private information of the user, leading to privacy loss [3]. Second, SSO users also face privacy threats from curious and colluding RPs [4], [5]. The identity tokens generated by an IdP for an individual user often contain common user identifiers, such as email, zip code, and name, enabling a group of RPs to perform *linkage attacks* to track and profile a user's behavior [6], [7]. Last but not least, the IdP also represents a *single point of failure*, since it is solely responsible for managing and verifying user login credentials. The IdP must always be online to authenticate users and issue identity tokens; any IdP service downtime would hinder its users from timely access to RP services [8], [9].

Recently, anonymous credential (AC) has emerged as a promising solution for enabling robust privacy-preserving SSO [10]–[13], particularly for addressing the linkability and single-point failure challenges. AC is a type of attribute-based credential that enables a user to prove certain identity attributes to a verifier without revealing unnecessary information. It has been studied extensively in the past [14]–[17] with emerging applications in privacy-preserving user/device authentication and identity management [18]. When AC is applied to SSO, a user typically obtains a digital credential $Cred$ containing a list of identity attributes along with a signature from the IdP. During a login session with an RP, the user generates a modified version (called a "presentation") of $Cred$ which selectively discloses $Cred$'s attributes as required by the RP; no unnecessary user information is disclosed. Crucially, the presentation contains a zero-knowledge proof (ZKP) that attests to the authenticity and integrity of the disclosed attributes (as inherited from a valid credential) while revealing no linkage to the original credential $Cred$ or other presentations of the same credential. The presentation can be independently verified by the RP, fulfilling user authentication without involving the IdP. This rules out the IdP for a single point of failure or user tracking.

While AC-based solutions offer appealing privacy benefits, they face outstanding challenges with respect to the support of essential SSO security functions and deployment efficiency.

Challenge-1: RP authentication vs. unlinkability. Existing AC-based SSO schemes lack support for RP authentication, which is nonetheless important for protecting users from illegal content and phishing scams by unauthorized RPs. In OIDC's specification [1], RP authentication is mandatory in the *Authorization Code Flow* (ACF), where the RP must authenticate to the IdP before proceeding to receive the authorization token and eventually the identity token. In particular, RP authentication requires a user to report the target RP to the IdP who keeps the list of all approved RPs. For AC-based SSO solutions, this would reintroduce the risk of user linkage since the verifier could trivially track the user's access requests to different RPs, forfeiting the unlinkability property once achieved through anonymous authentication. For this reason, existing AC-based SSO schemes [10], [11], [13] are only designed to support the *Implicit Flow* (IF) which does not require RP authentication nor any interaction between the RP and the IdP.[1] A practical solution should support RP authentication without creating new risks of user linkage.

Challenge-2: Efficiency and backward compatibility. AC-based SSO schemes commonly require the RP to undertake user authentication that typically involves verifying the zero-knowledge proof in a credential presentation. The verification workload is nontrivial compared to the existing practice in OIDC where the RP only needs to verify the public key signature in an identity token. This not only leads to performance bottlenecks when the RP has to handle millions of user requests in a short time frame but also, to some extent, defeats the purpose of *single sign-on* since each AC-based user-RP interaction constitutes a standalone authentication process [19]. To preserve an RP's operational efficiency, the new solution should not significantly increase an RP's verification workload. RPs should be able to continue using their legacy verification routines (i.e., verifying signatures in identity tokens) in the incumbent ACF workflow.

Recent privacy-preserving SSO proposals positioned as OIDC extensions have partially addressed the above challenges. AIF [12] extends the Implicit Flow with AC-based RP authentication so that the IdP does not know which exact RP is accessed by the user. This approach, however, requires RPs to get involved in AC management and presentation that brings non-trivial overhead. MISO [19] leverages a trusted execution environment (TEE) to establish a trusted identity mixing service to perform RP authentication for privacy-sensitive users. It nonetheless requires full trust in the TEE hardware for the mixing service, which may pose a new risk of single-point failure.

We introduce VeriSSO, a new AC-based privacy-preserving SSO protocol that supports RP authentication and is fully compatible with the incumbent OIDC's ACF workflow. For a consistent narrative, we adopt the World Wide Web Consortium's (W3C) Verifiable Credential (VC) terminology [20] to describe AC functionalities. In the registration phase, each user receives from its IdP a VC containing a list of attributes. In the authentication phase, the VC can be used to compute verifiable presentations (VPs) that selectively disclose the user's attributes along with ZKPs attesting to the VP's validity. The key novelty of VeriSSO is to employ an independent committee of *authentication servers* to provide RP authentication, VP-based user authentication, as well as identity token generation in a threshold manner. This preserves user unlinkability while allowing the RPs to continue operating in the incumbent ACF workflow and being robust to a single point of failure.

Binding RP Authentication with Anonymous User Authentication. To perform RP authentication without creating user linkage, we require a user to present the RP's authentication code $\mathsf{Auth}_{\mathsf{RP}}$ to every authentication server (instead of the IdP as in the incumbent ACF). The user then presents a VP to the authentication servers, disclosing the minimally required attributes per the RP. This two-step process essentially binds the RP authentication with VP-based anonymous user authentication. As long as the VP cannot be linked to a specific user, neither can $\mathsf{Auth}_{\mathsf{RP}}$. The authentication servers also cannot learn the user's real identity or RP accessing patterns, addressing Challenge-1. The IdP is not involved in any stage of the user authentication flow. Our VP design also supports *lawful de-anonymization* to hold a user accountable if anonymity is misused. The user identifier is encoded in the VP and can be decrypted by a threshold majority of authentication servers.

Threshold-based Generation of Authorization Code and Identity Token. After verifying the user VP, VeriSSO requires the committee of authentication servers to generate *shares* of the authorization code τ_{ac} through a multiparty computation (MPC) protocol so that the user can reconstruct τ_{ac} from a threshold fraction of the shares and delivers it to the RP. With τ_{ac}, the RP can then retrieve a share of the identity token τ_{id} from each authentication server, resembling a similar identity token retrieval process between the RP and IdP in the legacy ACF procedure. For the RP, τ_{id} can be reconstructed from a threshold fraction of the identity token shares, and then it can be used for signature verification as in the legacy procedure. Throughout the process, there is no single point of failure as long as the threshold fraction of authentication servers behave correctly.

In summary, we make the following contributions with VeriSSO:

- It is the first privacy-preserving SSO solution that fully supports RP authentication without introducing new risks of user linkage or single-point failures. This is achieved by binding RP authentication with AC-based user authentication with the help of an authentication committee.
- It allows RPs to continue their legacy user verification routine, i.e., verifying the user authentication success by the public key signature in an identity token. The added time cost, which is for reconstructing an identity token, is minimal (at the scale of milliseconds).
- We implemented a prototype of VeriSSO. The experiments show that a complete SSO authentication flow can finish within 100ms, demonstrating a reasonable time cost

[1]Implicit Flow was deprecated by OIDC due to the lack of RP authentication among other security risks, making ACF the predominant flow.

for a privacy-preserving solution.

II. Background

A. Single Sign-On Basics

In a common SSO setting, a user is prompted with a “login with SSO” option when requesting access to an RP’s web portal. The user is then redirected to its IdP for the actual authentication, which can involve normal login methods such as username-password or biometrics. Once authenticated, the IdP produces a signed **identity token** and delivers it to the RP. The RP extracts the user identity attributes, e.g., email and institution, from the identity token and verifies their legitimacy with the IdP’s signature. As a result, the RP does not need to authenticate the user independently, and the user does not need to keep separate login credentials for different RPs. SSO also enables a user to authorize resource access from across different providers. For example, a user may request an RP to download their profile data, images, credit score, etc, from remote servers. The user authorizes the host server using an **access token** that allows it to fetch the required data. In the actual SSO deployment, represented by OIDC, there exist two main types of workflow.

The **Implicit Flow** (IF) was developed around 2010 when mobile app platforms were new, and single-page apps were emerging. Browsers at that time did not support cross-domain POST requests, making it necessary to use the implicit flow for OAuth in browsers. The implicit flow omits the client secret and returns the access token directly in the redirect from the authorization server, simplifying the process by skipping the second step. The access token is returned in the redirect URL, which can be intercepted if the redirect is compromised. Examples of vulnerabilities include captive Wi-Fi portals intercepting redirects or browser extensions that can access the address bar and log URLs. Extensions can monitor and capture access tokens from the URL. Browser sync features (like Chrome syncing) can propagate access tokens across multiple devices, increasing the risk of theft.

The **Authorization Code Flow** (ACF) involves a two-step process (depicted in Figure 1) where a code is received and then exchanged for an access token, providing an extra layer of security. By not returning the access token directly in the redirect, the authorization code flow reduces the risk of interception during the initial request. Modern browsers now support cross-origin resource sharing (CORS) policies, enabling secure cross-domain POST requests needed for the authorization code flow. Proof Key for Code Exchange (PKCE) [21] was developed to enhance security, especially for public clients (e.g., mobile apps), by using a dynamically generated cryptographically random secret during the authorization process. This removes the need for a static client secret. The authorization code is exchanged in a server-to-server back channel, reducing the points where the access token can be intercepted. PKCE uses dynamically generated secrets for each authorization request, making it harder for attackers to use intercepted codes. With modern browser capabilities and the introduction of PKCE, the implicit flow is no longer necessary

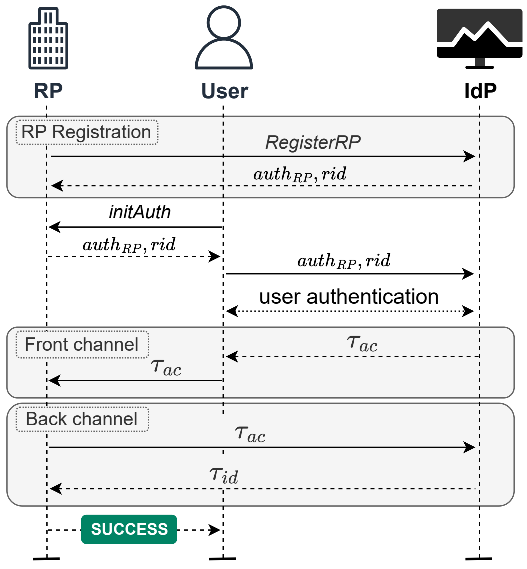


Fig. 1: OIDC’s Authorization Code Flow (ACF)

or recommended. The ACF with PKCE offers a more secure alternative [22], [23]. Due to the security risks of IF, ACF has evolved as the most widely used and recommended protocol flow for SSO in the developer community [24].

B. Verifiable Credentials

A Verifiable Credential (VC) is a digital credential document comprising a set of claims (also called attributes) about a user and a digital signature certifying the integrity of these claims [20], [25]. A VC is generated by an *issuer* who creates the digital signature with its private key and assigns it to the user. To users, VCs are analogous to everyday credentials like driver’s licenses or medical insurance cards and can be stored in digital wallets. A user can generate a Verifiable Presentation (VP) of attributes from one or more VCs which can be used to access a certain resource or service from a provider. The provider, also called a *verifier*, can verify the authenticity and integrity of the VP’s attributes before granting access. Both VCs and VPs are tamper-evident as they can be cryptographically verified.

Under the W3C’s Verifiable Credential framework [20], Anonymous Credential (AC) is a special type of VC scheme that protects users’ privacy, albeit it evolved independently in the past [14], [17], [26], [27]. AC enables *selective disclosure*—the VP generated from an AC can selectively disclose the AC’s attributes so that sensitive attributes like the user’s identifier or age are hidden from the verifier. Crucially, a zero-knowledge proof (ZKP) is used to attest to the integrity of the disclosed attributes in a VP. A ZKP is proof that the prover (i.e., user) knows a value (i.e., the undisclosed attributes in the original AC) without conveying any information apart from the fact that they know the value [28]. Therefore, the VP is sufficient for the verifier to authenticate the user. Moreover, VPs generated from the same AC are also *unlinkable* [29], further protecting the AC owner's anonymity from signature-based tracking. An AC scheme can be composed by a multi-message short signature scheme, such as the CL signatures [14], [15], [30] and BBS signatures [31]–[33], and an efficient ZKP scheme.

III. System Overview

A. Participant Model

We consider five types of participants in the SSO ecosystem: identity provider (IdP), relying party (RP), user, authentication server (AS), and verifiable data registry (VDR).

An **IdP** is an entity that maintains a trustworthy identity database of registered users and RPs. It is capable of generating public key signatures over a registered identity, attesting to its validity. Common IdPs include public web platforms (e.g., Google and Meta), universities,and specialized identity management services (e.g., Okta).

An **RP** offers online services/resources to users. In the SSO ecosystem, RPs register their service terms with an IdP, stating their own digital identities, resource URLs, and user attributes policy (i.e., what identity attributes are required for providing service to a user).

A **user** is a customer who wants to access online services/resources from RPs via the SSO option. Users have pre-existing registration with the IdP, stating their digital identity attributes.

An **AS** is capable of verifying the validity of users and RPs. The vanilla SSO standards defined by OIDC do not require a standalone AS since the IdP verifies users and RPs all by itself. However, in commercial space, standalone authentication services such as Auth0 [34] and OneLogin [35] begin to gain popularity due to their benefits in alleviating IdP workload and user convenience. Building on this intuition, VeriSSO relies on a **committee** of independent ASs to work in a decentralized fashion to fulfill user/RP authentication as well as identity token generation.

A **VDR** provides a public bulletin board for publicly accessible PKI information to all the participants. These may include the public key certificates of the RPs and IdPs and the list of blacklisted user credentials or users. VDR also provides the ASs with access to the IdP's Trusted Anchor List (TAL) where it publishes the metadata of registered RPs and their service terms.

B. System Goals

We aim for the following privacy, security, and practicality goals for our SSO solution.

Privacy Goals:

- **Minimal disclosure.** In each SSO session, a user only needs to disclose minimally required identity information to the RP and ASs, for instance, the association with a certain IdP. By hiding personally identifiable information from the disclosure, users can remain *anonymous* in SSO operations.
- **IdP/RP/AS unlinkability.** An SSO access attempt should be *unlinkable* by a curious IdP, RP, or AS to any specific user. Specifically, the IdP cannot decide which RPs a particular user has accessed; an RP or an AS cannot decide which specific user is making the access attempt among multiple access attempts.
- **Collusion resistance.** The above minimal disclosure and unlinkability properties should hold against any collusion between the IdP, RPs, and ASs.

Security Goals:

- **User authentication:** For an RP, only legitimate users registered with an IdP (with whom the RP has also registered) can sign on for service.
- **RP authentication:** Only legitimate RPs registered with an IdP can provide service to the IdP's users.
- **User accountability:** Anonymized/derived user credentials should be traceable and de-anonymized in case of misuse or lawful interception.
- **No single point failure:** The compromise of individual IdP/RP/AS cannot affect the integrity and availability of the SSO process in the user authentication phase. The scheme also does not have to use trusted hardware like TEE.

Practicality Goals:

- **Legacy-compatibility.** RPs should be allowed to continue their incumbent SSO routines as in OIDC's ACF workflow. An RP can verify the success of user authentication by checking the digital signature embedded in an identity token.
- **Efficiency.** The added cost for RPs and users should be lightweight. Particularly, RPs do not need to perform expensive zero-knowledge proof verification as in existing AC-based SSO solutions [10], [11], [13].

C. Threat Model

Below we describe the adversarial cases for each participant type considered in the design of VeriSSO.

IdP. We assume an IdP is *honest-but-curious* in that it will execute assigned routines faithfully but is motivated to track the user's access history to different RPs.

RP. A *malicious unregistered* RP may spoof users for the aim of phishing or scams. We also consider registered RPs to be curious about users' sensitive identity information. Multiple RPs may collude to discover the real user identity behind different SSO sessions.

User. Malicious unregistered users are motivated to obtain access to an RP. Similarly, a registered user may also wish to access RPs without passing the SSO authentication process.

AS. Individual ASs may not follow the designated routine due to either compromise or service outage. ASs are also curious about a user's access history, just like IdP and RP. However, we require a threshold fraction of the AS committee be honest (i.e., executing the designated routine correctly) and available. Moreover, an individual AS may collude with RPs (to help phish the users) or collude with the users (to help scam the RPs for service).

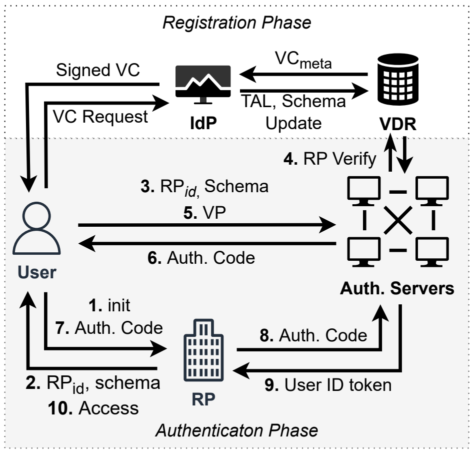


Fig. 2: VeriSSO Overview

D. *VeriSSO Overview*

VeriSSO is an authentication protocol that helps a user sign on for the resources or services provided by an RP. VeriSSO is based on the intuition that by binding RP verification with VP-based user verification, an AS is able to authenticate the RP (for its registered identity) and the user (for the validity of its VC) without learning the full identity profile of the user. The committee of ASs (i.e., the authentication committee) can then jointly issue an identity token that can be verified by the RP. Specifically, VeriSSO has two phases: the registration phase and the authentication phase. Figure 2 shows an overview of the VeriSSO workflow.

In the **registration phase**, users and RPs register their identities and service terms with the IdP, who are trusted organizations such as universities, government agencies such as the Department of Motor Vehicles (DMV), or online platforms such as Google [36]. A user obtains VCs from the IdP, with each VC consisting of several attributes of the user. The attributes are drawn from the IdP's internal records such as student databases, and/or physical identities such as driver's license or passport. The user is equipped with a digital wallet application (e.g., as a browser extension) capable of verifying and storing VCs issued by the IdP. The wallet can generate VPs based on a VC according to the user's interaction with an RP. The user VC request and issuance process is shown in the upper section of Figure 2. The RPs need to register their identities and service terms with the IdP as the condition to serve the IdP's registered users. A registered RP's identity and its associated terms/requirements are stored in the VDR as a metadata entry in the IdP's TAL, which is accessible by the ASs. We note that RP's metadata is usually in a specified format (like XML) in existing IdP's databases [37]. It contains several pieces of information such as entity descriptor (RP identifier and IdP identifier), service endpoints, public key certificates of IdP and RP, the expiration time of metadata, and the RP's contact information. In our system, storing RP metadata in the VDR allows the IdP to delegate the RP authentication task to the ASs.

In the **authentication phase**, the user proves its identity (using an RP-specific pseudonym) to an AS committee-verified RP and gets access to the resources in a privacy-preserving manner. As shown in the lower section of Figure 2, after the user initiates the SSO option (step ①), RP delivers its requirements and proof of its registration with the IdP (step ②) to the user. Then, the user relays the RP's information to the AS committee (step ③) who verifies the RP's claims with the TAL records on the VDR (step ④). Following the RP's requirements and TAL specifications, the user derives a VP by selectively disclosing attributes from its VC (step ⑤). The committee verifies the VP (step ④) and returns an authorization code to the user (step ⑥), who forwards it to the RP (step ⑦). Next, the RP establishes a secure back-channel with the committee nodes and exchanges the authorization code (step ⑧) for the user identity token (step ⑨). The identity token comprises the disclosed user attributes in a standard JSON Web Token (JWT) format with a threshold signature. Upon successful aggregation of the threshold signature and verification of JWT, the RP grants access to the user (step ⑩). Specifically, the generation of an authorization code (step ⑥) and identity token (step ⑨) leverages efficient threshold schemes to resist single point of failure, which we will show in section §V.

IV. Cryptographic Building Blocks

A. *BBS Signatures*

The Boneh-Boyen-Shacham (BBS) signature scheme [31] was originally proposed as a group signature scheme. BBS signature has been used as a building block for anonymous credentials and attestation schemes, represented by the schemes by Camenisch et al. [15], [17]. An extension of the BBS signature called BBS+ by Au et al. [33] slightly modifies the signature composition so that it is proved secure under the q-SDH assumption. Recently, Tessaro and Zhu [38] showed the security properties of the original BBS signature scheme [31] under the same q-SDH assumption and it has been adopted for standardization in the latest RFC draft [39]. We provide an outline of the BBS signature algorithm in Appendix A.

Threshold BBS Signature. In the context of anonymous credential issuance, the single issuer can be a single point of failure because it may become unavailable or corrupt leading to an inconsistent state of the credential issuance phase. To tackle this limitation, Doerner et al. [40] extended the signing protocol of the BBS+ signature scheme into a threshold multi-party signing protocol. The idea mainly leverages the concepts of threshold secret sharing and MPC protocols to distribute the anonymous credential issuance phase. The threshold BBS+ protocol can be broken down into the setup phase, shared signing phase, and aggregation phase.

In the setup phase, the MPC servers collectively generate a key pair. Each server commits to a set of $(L + 1)$ random scalars. The key generation functionality selects the secret key x and distributes it using Shamir's secret sharing algorithm.

The signing phase is initiated by a client who submits a signing request for L-messages to a certain number of MPC servers. At first, the servers use Lagrange coefficients to produce shares of the secret key x_i and select a secret blinding factor r_i. Then the servers collectively commit to random scalars (e, s) and engage in a pairwise multiplication protocol to secretly compute functions involving (r_i, x_i). The client combines results from J threshold number of servers in the aggregation phase to produce a valid BBS+ signature given by $\sigma = (A, e, s)$, where the aggregated element A is defined as:

$$A := \frac{\sum_{i\in J} R_i}{\sum_{i\in J} u_i} \; ; \; R_i = r_i\cdot(g_1 + \sum_{k\in[L]} m_k\cdot h_k); u_i = r_i\cdot(x+e) \tag{1}$$

This construction eliminates the single point of failure at the credential issuance phase with light overhead for signature aggregation, which is suitable for resource-constrained users.

In VeriSSO, we adapt the original BBS signature scheme [31], [39] into the threshold form in a similar construct, which we call the threshold BBS signature. Instead of thresholdizing credential issuance, the threshold BBS signature scheme is executed by the ASs (as the MPC server) to issue authorization codes and identity tokens to the RPs and users. This adaptation crucially allows RPs to use their legacy signature verification capability while supporting the efficient delivery of disclosed user attributes.

B. ZKP and Selective Disclosure

Zero-knowledge proofs (ZKP) are a general class of protocols between two parties, the prover and the verifier. Using a ZKP, the prover convinces the verifier of the validity of a given statement without revealing any information beyond the truth of the statement. The concept of zero-knowledge emerged in the 1980s [41]. Schnorr's identification protocol [42] is recognized as the first ZKP protocol. In this protocol, the prover can demonstrate knowledge of a secret $x \in \mathbb{Z}_p$ (discrete logarithm) to the verifier, corresponding to the prover's public key $h := g^x \mod p$. Originally, Schnorr's protocol was a Σ-protocol involving three rounds of interaction and assumed an honest verifier. By applying the Fiat-Shamir heuristic [43] under the Random Oracle Model (ROM) [44], it can be transformed into a non-interactive zero-knowledge protocol (NIZK) that is secure against arbitrary cheating verifiers [45].

Selective disclosure is an elegant feature of the BBS signature proposed initially by [33]. This allows the holder of a BBS signature to hide one or more messages while preserving its validity. This is achieved using a *signature proof of knowledge (SPoK)*, a type of ZKP, that allows a prover to disclose messages partially and at the same time randomize multiple presentations of the original signature.

In VeriSSO system architecture, we use the BBS signature algorithm to sign VCs and let users produce VP for authentication. To realize selective disclosure for VP, a SPoK is constructed using the protocol described below:

$\underline{BBS.SPoK:}$

- The prover has signature $\sigma = (A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$ and can compute $B = g_1 \prod_{i=1}^{L} h_i^{m_i}$ for a vector $\vec{m}$ of L messages.
- To selectively disclose messages $att_D = \{m_i\}_{i\in D}$ and hide $att_H = \{m_i\}_{i\notin D}$, select a random nonce $r \leftarrow \mathbb{Z}_p^*$.
- Randomize the original signature σ as $A' = A^r, \quad B' = B^r, \quad \bar{A} = (A^{-e}\cdot B)^r = (A')^{-e}\cdot B'$.

$\underline{BBS.VerifySPoK:}$

- The verifier receives $(att_D, A', \bar{A}, B', \pi_1, \pi_2)$ and verifies the bilinear equality: $e_b(A', w) \overset{?}{=} e_b\left(\bar{A}, g_2\right)$ and the proof Π given by:

$$\text{SPoK}\left\{(att_H, A, e, r) : \frac{\bar{A}}{B'} = \pi_1 \wedge g_1 \prod_{i\in D} h_i^{m_i} = \pi_2\right\} \tag{2}$$

C. Blind Signature

A blind signature is a cryptographic scheme in which the signer is not able to see the contents of the messages that are being signed. Like most standard digital signatures, a blind signature is unforgeable and can be verified against the signer's public key. *Strongly-blind signatures* are protocols in which the signer does not learn any useful information about the message or the signature, whereas in a *weakly-blind signature* it gets to know the signature. Most of the digital signature schemes based on RSA, ECDSA, BBS, etc. can be adapted to support blind signing by obscuring the message using a random factor commonly known as **blinding factor** [46]–[48]. In [40], the authors discuss a *weakly-blind* version of the BBS signature protocol, in which the subject or the messages are hidden from the signer. In VeriSSO, we leverage the partially blind signing properties of BBS signature scheme [38], [49], [50] to facilitate blind VC creation where the signature is issued to a commitment of the messages/attributes. The signature thus produced is never disclosed to the RPs during the authentication phase and is known to the IdP and user only.

V. Detailed Design

This section explains the VeriSSO protocol, with the detailed workflow illustrated in Figure 3.

A. Registration Phase

1) RP Registration: VeriSSO protocol requires RPs to register with the IdP in which they exchange important metadata with each other and store them in a TAL on the VDR. The registration process is crucial for establishing a trust relationship and ensures that legitimate RPs' identity and service policies can be appropriately verified. This is similar to the standard SSO schemes such as OIDC's Registration Protocol [51]. In VeriSSO, RPs provide their identity (such as public key, digital certificate, or name), service endpoints, and policies regarding the requirement of user information. The registration process is defined using the following functions:

- **RegisterRP**$(\boldsymbol{id_{RP}}, \boldsymbol{meta_{RP}}, \boldsymbol{sch_{VP}}) \rightarrow (\boldsymbol{rid}, \boldsymbol{meta_{IdP}})$: The RP invokes the registration with the IdP with its identity information id_{RP}, metadata $meta_{RP}$

(containing its service endpoint url_{RP}), and VP attribute schema sch_{VP} (i.e., the list of user attributes required for log-in). Upon successful registration, the IdP returns its metadata information $meta_{IdP}$ and a unique identifier rid as a token of its TAL membership.

- **AssignRID**$(\boldsymbol{id_{RP}}, \boldsymbol{meta_{RP}}, \boldsymbol{sch_{VP}}) \rightarrow \boldsymbol{rid}$: This function is called by the IdP as a subroutine of $RegisterRP$. The IdP commits the parameters into its TAL on the VDR and receives a rid for the RP.

2) User Registration and Credential Issuance: We assume users have a membership with the IdP and have verified their identity information using physical (e.g. Driver's License or Social Security) or institutional affiliations (e.g. university or workplace ID).

A user (as a client of an IdP) requests a VC from the IdP. The user selects a set of common attributes (e.g., profession, institution, student status, zip code, etc.) and submits a credential signing request. For generality, we denote $\vec{m} = \{m_1, m_2, \ldots, m_L\}$ to denote L attributes in the request and L is user-determined. We describe the credential request using the following function:

- **RequestVC**$(\vec{\boldsymbol{m}}, \boldsymbol{u}) \rightarrow (\boldsymbol{cred}, \boldsymbol{\gamma}, \boldsymbol{\sigma})$: The user selects a vector of messages $\vec{m}$ consisting of L attributes and submits it to the issuance endpoint of the IdP. In addition, the user binds its attributes to its device (where the wallet resides) using a local secret u, which is an extra attribute aside from $\vec{m}$. Then, the user wallet blinds $\vec{m}$ and u using a blinding factor d, i.e., $m'_j = m_j^d, \forall j = 1, ..., L$ and $u' = u^d$. The wallet is expected to receive a VC $cred$, a blind signature $\hat{\sigma}$ of the VC, and an IdP-assigned unique credential identifier γ (included in $cred$). The $cred$ generation is shown right below.

The IdP is responsible for issuing VCs to users by attesting a set of requested attributes in $\vec{m}$. It also ensures that each VC includes a unique identifier γ that acts as a backdoor mechanism for accountability of dishonest users. The user can hide the identifier during the authentication phase to preserve privacy and anonymity. We describe credential issuance using the following functions:

- **IssueVC**$(\vec{\boldsymbol{m}}', \boldsymbol{u}') \rightarrow (\boldsymbol{cred})$: The IdP first assigns a unique identifier γ. Next, it produces a W3C-standardized VC, $cred$, with the vector of blinded parameters $\vec{m}', u'$, and γ as the subject attributes. In the final step, the IdP produces a blind signature $\hat{\sigma}$ of the messages $(\vec{m}', u', \gamma)$ using the BBS signing algorithm described in section §IV-A.
  Specifically, on the input of key pair (sk, pk), $\vec{M} = (\vec{m}', u', \gamma)$, and a random scalar e, the IdP produces a blind signature $\hat{\sigma} = (A, e)$, where $A = \left(g_1 \prod_{i=1}^{L+2} h_i^{M_i}\right)^{\frac{1}{e+x}}$. The IdP then appends some auxiliary metadata and composes the VC as $cred = \{metadata,\ \vec{M}, \hat{\sigma}\}$. Upon completion of the protocol, $cred$ is sent back to the user.

B. Authentication Phase

The credential wallet on the user side stores the VC $cred$ received from IdP and uses it for authentication purposes. Figure 3 depicts an overview of the algorithms involved in user authentication. Next, we explain the individual steps involved in the process.

1) RP and User Authentication by ASs: These stages rely on the ASs to authenticate both the RP and the user in an SSO session. It contains the following functions:

- **InitAuth**$(\boldsymbol{url_{RP}}, \boldsymbol{id_{IdP}}) \rightarrow (\boldsymbol{rid}, \boldsymbol{sch_{VP}})$: The user initializes an SSO authentication session by accessing url_{RP}, the designated login web address/endpoint of an RP. It informs the RP of its identity provider, url_{IdP}, and requests for its membership token rid and schema definition sch_{VP}.
- **VerifyRP**$(\boldsymbol{url_{RP}}, \boldsymbol{rid}, \boldsymbol{sch_{VP}})$: To verify the authorization of the RP and its membership agreement with the IdP, the user queries the ASs to confirm the url_{RP}, rid, and sch_{VP} in the TAL of the IdP. When the user receives a threshold number of positive responses from the ASs, the RP is considered to be verified.
- **PrepareTempId**$(\boldsymbol{u}, \boldsymbol{\gamma}, \boldsymbol{domain_{RP}}) \rightarrow \boldsymbol{\phi}$: The user hides the permanent identifier γ and user secret u and generates a pseudonym for each RP. In this process, γ is the ciphertext generated with the ElGamal encryption [52], and u is used to produce an RP-specific pseudonym $\tilde{u}$. Finally, the user produces a proof ϕ, of his knowledge of (γ, u).
  Specifically, $\tilde{u}$ is generated as follows. On the input of user secret u and RP domain $domain_{RP}$ (from $meta_{RP}$), pseudonym $\tilde{u}$ is produced as:
  $$\tilde{u} = \tilde{h}^u; \tilde{h} \leftarrow H(domain_{RP}) \tag{3}$$
  where H is a cryptographically secure hash function. The user attaches a ZKP of his possession of the secret identifiers γ and u:
  $$\phi = \{ENC_{EG}(h^\gamma) \wedge \tilde{u} = \tilde{h}^u\} \tag{4}$$
  where $ENC_{EG}(h^\gamma) = (g^\alpha, y^\alpha h^\gamma)$ is an ElGamal encryption of the permanent identifier γ. Elements g, h are generators of $\mathbb{G}_1$, α is a random scalar, and y is the aggregated key of a threshold number of decryption authorities (which are chosen from the ASs in our system).
- **PrepareVP**$(\boldsymbol{cred}, \boldsymbol{sch_{VP}}) \rightarrow \boldsymbol{pres}$: The client reviews the attribute requirement from the RP's (as in schema sch_{VP}) and selects the attributes to disclose $\{m_i\}_{i \in D}$, and hides the remaining $\{m_i\}_{i \notin D}$. Next, the user prepares a signature proof of knowledge $SPoK$ to prove the validity of the signature σ as follows:
  On the input of VC signature $\sigma = (A, e)$, disclosed attributes $\{m_i\}_{i \in D}$ and hidden attributes $\{m_i\}_{i \notin D}$, the user computes $\pi_1 = A'^{-e}$, $\quad \pi_2 = B \prod_{i \notin D} h_i^{-m_i}$, where $B = g_1 \prod_{i=1}^{L} h_i^{m_i}$. A VP is composed as: $pres = \{meta_{VP}, \{m\}_{i \notin D}, A', B', \bar{A}, \pi_1, \pi_2, \tilde{u}, \phi\}$
- **VerifyVP**$(\boldsymbol{pres})$: Once an AS receives the $pres$, the

		RP Registration Phase		
RP		**IdP**		**VDR**
$req := (id_{RP}, meta_{RP}, sch_{VP})$	$\xrightarrow{req}$	$rid \leftarrow AssignRID(req)$	$\xrightarrow{req}$	$rid \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
$res \leftarrow RegisterRP(req)$		$res := (rid, meta_{IdP})$		$Commit_{TAL}(rid, req)$
$rid, meta_{IdP} := res$	$\xleftarrow{res}$	$return\ res$	$\xleftarrow{rid}$	$return\ rid$

		User Registration Phase (Credential Issuance)
User/Wallet		**IdP**
$u, d \overset{\$}{\leftarrow} \{0,1\}^{\lambda},\ \vec{m} := \{m_1, \ldots, m_L\}$		
$\underline{RequestVC(\vec{m}, u):}$		
$u' \leftarrow Blind(u, d)$		$\underline{IssueVC(\vec{m}', u'):}$
$\vec{m}' \leftarrow Blind(\vec{m}, d)$		$\gamma \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
$cred \leftarrow IssueVC(u', \vec{m}')$		$\vec{M} := \{\vec{m}', u', \gamma\}$
$(\vec{M}, \hat{\sigma}) := cred$	$\xrightarrow{u', \vec{m}'}$	$\hat{\sigma} := BBS.Sign(\vec{M}, sk)^{\dagger}$
$BBS.VerifySign(\vec{M}, \hat{\sigma}, pk)$		$cred := \{metadata, \vec{M}, \hat{\sigma}\}$
$\sigma \leftarrow Unblind(\vec{m}', u', \hat{\sigma}, d)$	$\xleftarrow{cred}$	$RegisterUser(cred)$
$VC := \{metadata, \vec{m}, u, \gamma, \sigma\}$		$return\ cred$

		Authentication Phase (ID Token Issuance)		
User/Wallet		**RP**		**AS**
$rid, sch_{VP} \leftarrow InitAuth(url_{RP}, id_{IdP})$		$\underline{InitAuth(url_{RP}, id_{IdP})}$		$\underline{VerifyRP(url_{RP}, rid, sch_{VP}):}$
$VerifyRP(url_{RP}, rid, sch_{VP})$	$\xrightarrow{url_{RP}, id_{IdP}}$	$res := (rid, sch_{VP})$		$\perp \leftarrow Check_{VDR}(rid, sch_{VP})$
	$\xleftarrow{res}$	$return\ res$		$return\ \perp$
$\underline{PrepareTempId(u, \gamma, domain_{RP}):}$		$\xrightarrow{rid, sch_{VP}}$		
$\tilde{h} \leftarrow H(domain_{RP})$		$\xleftarrow{\perp}$		$\underline{VerifyVP(pres):}$
$\tilde{u} := \tilde{h}^{u}$				$\perp \leftarrow BBS.VerifySPoK(pres)$
$\langle g, h \rangle \in \mathbb{G}_1$		$\xrightarrow{pres}$		$return\ \perp$
$\phi := ZKP\{ENC_{EG}(h^{\gamma}) \wedge (\tilde{u} = \tilde{h}^{u})\}$		$\xleftarrow{\perp}$		
				$\underline{IssueAuthCode(rid, \tilde{u}):}$
$\underline{PrepareVP(cred, sch_{VP}):}$		$\xrightarrow{rid, \tilde{u}}$		$token_{ac} \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
$att_D := \{m_i\}_{i \in D},\ att_H := \{m_i\}_{i \notin D}$		$\xleftarrow{\tau ac}$		$return\ SS(token \Vert rid)$
$\Pi \leftarrow BBS.SPoK(att_D, att_H, \sigma)$		$\underline{FinishAuth(\tilde{u}, \tau_{ac}):}$		
$pres := \{meta_{VP}, att_D, \tilde{u}, \phi, \Pi\}$		$req := (rid, \tilde{u}, \tau_{ac})$		$\underline{IssueIdToken(rid, \tilde{u}, \tau_{ac}):}$
$VerifyVP(pres)$	$\xrightarrow{\tilde{u}, \tau_{ac}}$	$\tau_{id} \leftarrow IssueIdToken(req)$	$\xrightarrow{req}$	$(\tilde{u}, att_D) := pres$
$\tau_{ac} \leftarrow IssueAuthCode(rid, \tilde{u})$	$\xleftarrow{\perp}$	$\perp \leftarrow BBS.VerifySign(\tau_{id}, pk)^{\ddagger}$	$\xleftarrow{\tau id}$	$\tau_{id} := \{rid, \tilde{u}, att_D\}$
$FinishAuth(\tilde{u}, \tau_{ac})$		$return\ \perp$		$return\ BBS.Sign(\tau_{id}, sk)^{\ddagger}$

Fig. 3: VeriSSO Protocol. $\perp$-Function call result as Success/Failure. †-Blind VC signing/issuance. ‡-Threshold BBS protocol (details omitted here, see function descriptions in §V-B). ▨: User-AS communication. Wrapper functions: H- Secure hash function. $Commit_{TAL}$- Update TAL records in VDR. $Blind$- Compute a commitment to the given input. $Unblind$-Reveal the commitment. $RegisterUser$-Update IdP's user database records. $Check_{VDR}$-Checks the TAL records for the given data. SS-Generate secret shares of a given input. ZKP-Zero-knowledge proof.

validity of its signature σ and undisclosed attributes can be verified according to the steps described in the function $BBS.VerifySPoK$ (equation 2).

2) Threshold ID Token Issuance and Verification: After verification of the presentation, the AS committee responds to the user with a valid authorization code τ_{ac}. The user forwards τ_{ac} to the RP as a notification of RP verification and user authentication. Finally, the RP requests the AS committee to release the user attributes in an identity token τ_{id} by supplying τ_{ac}. The process contains the following functions:

- **IssueAuthCode**$(\boldsymbol{rid}, \boldsymbol{\tilde{u}}) \rightarrow (\boldsymbol{\tau_{ac}})$: Once $pres$ and its $SPoK$ are verified, the user queries the AS committee for an authorization code for the specific session and rid and receives τ_{ac} as secret shares. In this process, the ASs interact and each generates and returns a secret share of τ_{ac}. The user can reconstruct the τ_{ac} by collecting a threshold number of the shares and then relays it to RP.
- **IssueIdToken**$(\boldsymbol{rid}, \boldsymbol{\tilde{u}}, \boldsymbol{\tau_{ac}}) \rightarrow (\boldsymbol{\tau_{id}})$: Upon receipt of the authorization code τ_{ac}, the RP is convinced of a successful RP verification and user authentication. Now, the RP requests the AS committee to release the user attributes disclosed in the VP by posting the authorization code τ_{ac} and its signature to AS committee. The AS committee invoke the threshold BBS signature protocol (as in §IV-A) to generate the shares of the BBS signature. It works as follows. The RP initiates the signing protocol by sending requests to t ASs who engage in an MPC protocol to generate a token consisting of the disclosed

user attributes and a threshold BBS signature given by $\sigma = (A, e)$. Each AS of the J signing parties, returns R_i and u_i, where $R_i = r_i \cdot (g_1 + \sum_{k\in[L]} m_k \cdot h_k)$ and $u_i = r_i \cdot (x + e)$. And x is the secret key, e is a random scalar, r_i is a secret share of a random scalar r, and L is the number of messages to be signed.
- $\mathbf{FinishAuth}(\tilde{\mathbf{u}}, \tau_{\mathbf{ac}})$: From the signature shares $(\{R_i, u_i\}_{i\in J})$, the RP can reconstruct the BBS signature of the identity token τ_{id} by perform the aggregation: $A := \frac{\sum_{i\in J} R_i}{\sum_{i\in J} u_i}$. Finally, the RP verifies the aggregated signature A with the AS committee's threshold public key (stored on the VDR). If verified, the RP grants access to the user and concludes the SSO session.

3) Lawful De-anonymization: A user does not reveal any unique identifier (signature, VC permanent identifier, or user secret) in a VP construction. During the sign-on process, the user prepares an RP-specific pseudonym $\tilde{u}$ to uniquely identify itself to an RP. If an anonymous VP must be de-anonymized for accountability, the AS committee can reveal the associated user. Certain AS members with lawful authorities (i.e., the decryption authorities) collectively use their private key (corresponding to the combined encryption key y) to decrypt the VC identifier ($ENC_{ElGamal}(h^\gamma)$) attached in $pres$. The AS committee will thus recover h^γ which can be traced back to the IdP assigned permanent identifier γ and to the user's VC.

VI. Security Analysis and Discussion

A. Security Analysis

Theorem 1 (Correctness of Authentication): If the q-Strong Diffie-Hellman (q-SDH) Assumption, Decisional Diffie-Hellman (DDH) Assumption, and Strong Computational Diffie-Hellman Inversion (SCDHI) problems, as outlined in [27], [53] (defined in Appendix C), hold and the zero-knowledge proof systems satisfy *completeness, soundness and zero-knowledge* properties, then VeriSSO guarantees user authentication, RP authentication, as well as user accountability.

Proof Sketch. The correctness of our SSO scheme essentially builds on its composing protocols, namely the BBS signature-based VC scheme, the threshold BBS signature for ID token generation, as well as the public key signature scheme for verifying RP authenticity. The BBS signature is known to be secure under DDH and SCDHI, while the threshold BBS scheme produces the same signature. The user accountability is provided by the completeness and soundness of ZKP, which proves the validity of the ElGamal encryption (which itself is semantically secure under DDH [52]) of the user's credential identifier γ.

Theorem 2 (Unforgeability): In VeriSSO's authentication phase, the VP generation and verification algorithms (see section V-B) are (t, Q, ϵ)-unforgeable for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$, that can make Q random oracle queries in time t and ϵ is a negligible value.

Proof Sketch. The unforgeability property can be derived intuitively from the unforgeability of the BBS signature scheme [31] It ensures that an adversary $\mathcal{A}$, cannot forge a legitimate credential $Cred$ or a valid proof of a valid attribute set $\vec{m}$ signed by the issuer's signing key sk. A $Pres$ in VeriSSO comprises the ZKP of the issuer's signature σ (known only to the user and IdP), IdP assigned permanent identifier γ (known only to the user and IdP), and user device secret u (known only to the user). Forging a $Pres$ would thus require full control of the wallet or user's device.

Theorem 3 (Unlinkability): The credential scheme in VeriSSO satisfies unlinkability and anonymity across different RP access sessions for all PPT adversaries $\mathcal{A}$. Given the proof, $\mathcal{A}$ can simulate the presentation $Pres$ only if the complete set of attributes $\vec{m}$ in the credential and the issuer's signing key sk are known.

Proof Sketch. If anonymity in an AC scheme is maintained across multiple presentations of the credential, it is considered to meet the criteria of multi-show unlinkability, and the same can prove these two properties for VeriSSO. In the authentication phase, the user hides the VC signature σ and the permanent identifier assigned by IdP γ and the device secret u, and presents a ZKP of their knowledge and possession. Moreover, the user derives a new pseudonym $\tilde{u}$ to produce a unique identity for each RP.

Remark on collusion resistance. During the user authentication phase of the VeriSSO protocol the user partially discloses the VC attributes. While drafting the VP, it obscures the permanent identifiers (u, γ) and derives a ZKP of their possession. In addition, each VP is derived with an RP-specific pseudonym as the main identifier. Together, this construction allows the protocol to provide collision resistance.

B. Discussion on Practical Deployment

We discuss VeriSSO's design choices and practicality.

Who will be the authentication servers? Using third-party authentication services to facilitate client access control has been quite popular in the commercial space, represented Auth0 [34] (now a subsidiary of Okta) and OneLogin [35]. In VeriSSO, we envision that these commercial authentication services can be transitioned into the authentication committee, given that they can communicate for joint verification tasks. New candidates for the authentication committee include delegates of well-known IdPs and RPs. The profit model and incentive mechanisms of authentication servers are an important but orthogonal research effort.

What if a user's credential wallet is exposed or stolen? According to the design of VeriSSO, every credential is linked to a permanent identifier assigned by the IdP. In the event of a wallet theft or compromise, the user can instruct the IdP to terminate the credential and publish an alert to the VDR. In this way, the authentication committee will not approve any VPs produced using the reported credential. However, a more rigorous credential revocation scheme should be a direction for future improvement.

Support for Multi-Factor Authentication. VeriSSO is an authentication protocol leveraging the single factor of credential ownership. However, an MFA module could be easily integrated to offer multiple levels of security using standard protocols like FIDO [54]. We consider MFA support to be an implementation design decision, and VeriSSO architecture can accommodate such an extension while preserving its privacy and usability properties.

Supporting Multiple IdPs. Some RPs may require users to prove attributes from different IdPs, for instance, for showing diploma proof issued by a university and completion certificates issued by an online course provider. The W3C standard's VC and VP data model, which we used intrinsically, supports multiple issuers [36] and allows a user to compose a VP using multiple VCs.

Potential Usage of Interactive ZKP. The VeriSSO protocol requires several interactions between the user and the Authentication Committee for RP and VP verification. This provides a scope for the possible use of $\Sigma - protocol$ for proof generation to improve the overall performance of the front channel subroutines. We will explore this direction for future extensions.

Colluding RPs and/or IdPs. The VP derived during the authentication phase is attributed with an RP-specific pseudonym instead of revealing the permanent VC identifiers (u, γ) to prevent direct user linkage. However, it is possible that the attributes disclosed by multiple VPs may be engineered to track users. That would require colluding parties to remain compromised and study user activity logs for a significant amount of time.

VII. Implementation and Evaluation

A. Prototype Implementation

We implemented a prototype to evaluate the core functionalities of VeriSSO: (1) issuance and verification of VC, (2) proof generation and verification of VP, and (3) issuance, aggregation, and verification of threshold identity token.

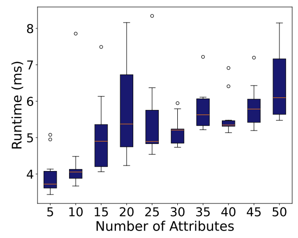
Our implementation of BBS signature uses BLS12-381 Elliptic curve [55]. The various core subroutines of VeriSSO use the Rust library bbs_plus version 0.23.0 [56] to implement the standard and MPC versions of the signature scheme according to the protocol discussed in Section IV-A.

Our experimental setup includes a server for simulating *IdP* functionalities for the setup and user registration phase. In the authentication phase, the user interacts with *AS* nodes running server programs for managing threshold tokens. We deploy our experimental arrangement on a desktop PC equipped with a 12th Gen Intel Core i7 processor (12 cores, 2.10 GHz), 32 GB DDR5 RAM, and Ubuntu 22.04 LTS operating system. For simplicity of implementation, the *AS committee* nodes are logical instances running within the same environment in multiple threads. Our experiments use a total of 8 AS nodes with a threshold requirement of 5 nodes. For scalability experiments, we vary the number of ASs from 2 to 10 within the hardware capabilities of our experimental setup.

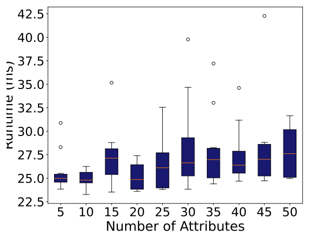
B. Evaluation Results

This section presents our findings on the experimental evaluation of VeriSSO prototype. Overall, we observe an execution time constraint in comparison to established frameworks like OpenID, due to the involvement of advanced cryptographic schemes. The time costs increase with the size of the credential (number of attributes) as it requires more group operations. However, credentials with a large number of attributes are less common, and a moderate attribute set credential (˜15 attributes) completes a user sign-on in less than 100ms.

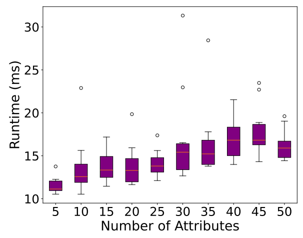
Next, we present the experimental results from the main individual functionalities of VeriSSO system design.



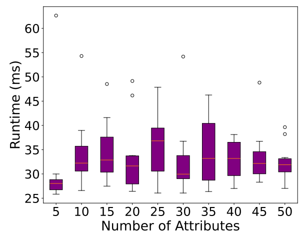
(a) Credential Issuance



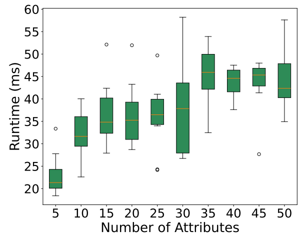
(b) Credential Verification



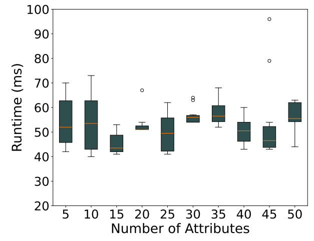
(c) Proof Generation



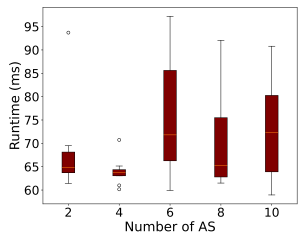
(d) Proof Verification



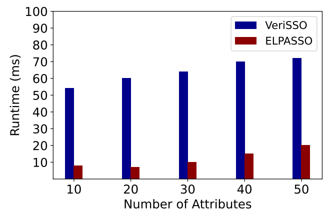
(e) Token Generation



(f) Token Verification



(g) AS Scalability



(h) Runtime Comparison

Fig. 4: Runtime cost for issuance (4a) and verification (4b) of credential in the setup phase. Runtime cost for presentation (4c, 4d) and ID token processing (4e, 4f) in the authentication phase. Impact of AS scalability in the protocol runtime (4g). Comparison of the overall execution time of related SSO protocols (4h).

User Registration. In Figure 4a and 4b, we demonstrate the execution time for user registration in the setup phase.

The *issuance* time (Figure 4a) indicates the time taken by the IdP to produce a BBS signature for a given number of user/subject attributes. Upon receipt of a VC signature, a user performs a signature *verification* (Figure 4b) before storing it. Together, this is a one-time setup for a user and ranges in milliseconds from a smaller number of attributes to higher values as attributes increase. The results indicate a linear increasing relationship with the number of attributes and a constant difference (~1 ms for issuance and ~2.5 ms for verification) between the issuance and verification times.

User Authentication. In our experiment for the authentication phase, we capture the execution time for *proof generation* (selective disclosure of half attributes and signature hiding) on the user client and *verification* by at least one committee node. This is the runtime of the front channel processes in VeriSSO protocol and covers a significant portion of the user-sign on delay. Figure 4c shows linearly increasing runtime with the number of user attributes. It is also observed that for a smaller number of attributes, the difference between proof generation (Figure 4c) and verification time (Figure 4d) is minimal, whereas proof verification takes ~2$\times$ the time of proof generation for a larger attribute set (35-50 attributes).

Threshold ID Token Issuance, Aggregation, Verification. We present the experimental results of threshold token issuance in Figure 4e, aggregation, and verification in Figure 4f. We observe an incremental pattern with attribute count, though the difference is small. The median values range from 20ms to 45ms for token *generation* (Figure 4e) and 40ms and 50ms for token *aggregation and verification* (Figure 4f). In general, the execution time of the threshold signature is significantly shorter than the standard BBS signature (as shown in Figure 4a and 4b) because of the workload distribution among the MPC nodes. The threshold token is issued by the *AS Committee*, then aggregated and verified by the RP.

Performance Comparison and Scalability Analysis. Figure 4g shows how runtime scales with the number of ASs in a distributed environment. The runtime increased linearly with the number of servers. This trend is likely attributed to the increased communication overhead and synchronization required between servers in a distributed setup. Nevertheless, the experiment demonstrates the predictability of scalability, which is crucial for environments that necessitate distributed authentication. The comparison of VeriSSO and EL PASSO (Figure 4h) under identical conditions reveals an increase in runtime with the growing number of attributes for both systems. Notably, VeriSSO underperforms compared to EL PASSO across all configurations. This is primarily because EL PASSO is based on the Implicit Flow that requires only two rounds of interaction between the user and RP and does not support RP authentication. VeriSSO follows the ACF sequence and involves ASs to verify VP and issue ID tokens that require five rounds.

VIII. Related Work

Various works have addressed user privacy in SSO schemes. SPRESSO [57] is an SSO protocol that aims to improve user privacy by making sign-in sessions on one SP indistinguishable from another. Lin et al. [58] propose a user-controlled SSO mechanism where users can generate a session key to establish a secure communication window for accessing different telemedicine providers. UPPRESSO [7] uses temporary pseudo-identity to access RPs in order to facilitate a privacy-preserving SSO experience, reducing inter-RP linkability.

Relevant to our work, [10], [59], [60] explored the possibilities of incorporating self-sovereign identity (SSI) and VC into SSO and federated identity management. Lux et al. [59] envision using SSI and VC to give more user control over personal data in the existing OIDC-based SSO where the IdP is still needed for VC verification in an online fashion. In comparison, we seek to minimize IdP's involvement in the authentication (i.e., not required to be always online) to address the availability issue. PRIMA [10] also discusses the limitations of OpenID in terms of IdP being able to log interactions between users and RPs. The authors suggest an identity management scheme reminiscent of standard IF with credentials instead of JWT tokens as in OIDC. This mechanism hides the RP's identity but lacks RP verification and requires IdP to be available to sign the credentials for each authentication attempt.

EL PASSO [11] was the first SSO solution leveraging anonymous credentials to achieve inter-RP unlinkability and lawful de-anonymization. It further achieves intra-RP linkability to reduce the Sybil risks, however, at the cost of RP-IdP collusion attacks when the IdP is also compromised. EL PASSO highlights the privacy issues of traditional SSO schemes such as OpenID Connect due to a tightly coupled IdP-centric protocol flow. As a result, IdP and RP can collude to infer user activity and violate purpose limitations (EU GDPR [61]) of the user agreement.

MISO [19] leverages Trusted Execution Environment (TEE) to establish a trustworthy intermediary to facilitate an anonymous authentication process while preserving the original SSO workflow. The TEE-based Mixer serves as a trusted IdP to RP and as an anonymous RP to IdP. This design allows IdP and RP to function without knowledge of their actual identities. In practice, the security provided by TEE hardware may not be available, and the mixing service itself may pose a new risk of single-point failure.

In comparison, we introduce a distributed authentication server committee to realize the privacy and availability features without relying on trusted hardware.

AIF [12] addresses similar privacy issues of SSO by extending the IF using blinded tokens. The authors emphasize the risks of identity exposure at IdP and the lack of RP authentication in standard IF design. The design of AIF leverages AC-based anonymous authentication for RPs as well (i.e., similar to AC-based user authentication). This approach, however, requires an RP to be involved in credential issuance and revocation steps. Privacy OIDC (POIDC) [62] addresses these issues using cryptographic commitments on RP identifier. However, the commitment signature in this approach does not prevent IdP from linking the user's access to different RPs. OPPID [64]

Related SSO Solutions \ Security/ Privacy Properties	No trusted hardware	No single-point failure	Inter-RP unlinkability	Intra-RP linkability	Selective disclosure	Lawful de-anonymiz.	Multi-device support	RP authentication	Compatible w/ OIDC ACF
SPRESSO [57]	✓[1]	✗	✗	✓[3]	✗	✗[5]	✓[6]	✗	✗
PRIMA [10]	✓	✗	✓	✓[3]	✓	✗	✗	✗	✗
POIDC [62]	✓	✗	✗	✓[3]	✗	✗[5]	✓[6]	✗[7]	✗
UPRESSO [7]	✓	✗	✓	✓[3]	✓	✗[5]	✓[6]	✗	✓
EL PASSO [11]	✓	✗	✓	✓	✓	✓	✓	✗	✗
MISO [19]	✗	✗[2]	✗	✓[3]	✗	✗[5]	✗	✓	✓
AIF [12]	✓	✗	✗	✓[3]	✗	✗[5]	✓[6]	✓[7]	✗
ARPSSO [63]	✓	✗	✗	✓[3]	✗[4]	✗[5]	✗[6]	✓	✓
OPPID [64]	✓	✗	✓	✓	✗[4]	✗[5]	✗[6]	✓[7]	✗
PESTO [65]	✓	✓	✗	✓[3]	✗[4]	✗[5]	✓[6]	✗[7]	✗
PASTA [66]	✓	✓	✗	✓[3]	✗[4]	✗[5]	✓[6]	✗[7]	✗
VeriSSO (this work)	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE I: Comparison of Privacy-preserving SSO Solutions. [1] SPRESSO is dependent on a third-party agent/server for delivering scripts and as an intermediary between RP and IdP. [2] MiSO eliminates single point of failure at IdPs, but relies on another single entity with trusted hardware. [3] User identity is linked to a fixed attribute like email (not pseudonymous). [4] Attributes disclosure is controlled by IdP. [5] Solutions do not consider anonymization or AC. [6] User identity/credential not linked to a physical device. [7] Solutions are based on SSO Implicit Flow.

extends the AIF protocol and adds user unlinkability by using pseudonyms computed by a variant of HashDH PRF [67]. AIF, POIDC, and OPPID primarily focus on privacy-preserving IF, which OIDC proposed for limited resource or non-PKI entities. Similar to our approach, ARPSSO [63] resolves these issues for the most widely adopted ACF model using anonymous credentials for RP authentication. In comparison, our design uses VC to facilitate user-centric identity management while satisfying the aforementioned properties.

PASTA [66] aims to improve the process of authorization token issuance from a single IdP by distributing its role among multiple servers using threshold oblivious pseudo-random function and token generation schemes. The key idea is to secret share the password hash and the master key used for token issuance using a threshold protocol to mitigate client impersonation and compromised servers. While PASTA is a good solution for distributed SSO, it lacks proactive security for recovering from a compromised state. PESTO [65] addresses this limitation and incorporates a recovery scheme that allows servers to reboot to a consistent state. Unlike PASTA, which requires servers to obtain dedicated keys from each client, PESTO derives key material using a proactively secure PRF. Both schemes use a distributed version of the RSA algorithm for token signing. However, both protocols are limited to the SSO Implicit Flow and lack NIST-recommended user anonymization [68] and RP hiding.

Parallel to the above schemes, the OpenID Foundation is also working on VC-based specifications for several use cases, including VC-based authentication and authorization, OAuth extension using VP, and self-issued credentials. The standardization draft [69], [70] under the OpenID Connect Working Group acknowledges the limitations and challenges of the existing OIDC model revolving around a central IdP and leverages the barebone W3C VC architecture in conjunction with OIDC sub-protocols like OAuth. OpenID describes VCs as a paradigm shift from centralized dependency on IdPs with privacy risks and lack of user control, and transformation to VCs offers an opportunity for user-centric identity management with decentralized control, enhanced privacy, and portability [70], which also partly motivated our work.

An overview of the comparison of various security and privacy properties of related privacy-preserving SSO solutions is presented in Table I. Please refer to Appendix D for more related work on general SSO security issues.

IX. Conclusion

We present VeriSSO, a new SSO protocol that leverages threshold signatures and VC-based distributed authentication to achieve user privacy and backward compatibility. VeriSSO allows users to control their information disclosure in SSO sessions while preserving user experience like traditional methods. Unlike other related credential-based SSO schemes, VeriSSO crucially supports RP authentication and requires minimal architectural changes on the RP side, providing easier compatibility and migration from existing solutions. We introduce a distributed authentication committee to take over the responsibilities of the centralized IdP. This design enables us to handle single points of failure at IdP, malicious RP access, RP-IdP collusion, credential verification, and threshold access token issuance without relying on trusted hardware. Moreover, our authentication protocol provides an identity backdoor for accountability and lawful de-anonymization. The experiments show that a complete VeriSSO authentication flow can finish within 100ms, demonstrating feasibility in practical use.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant number 2247561.

References

[1] O. Foundation, "What are openid specifications," https://openid.net/developers/specs/, 2023, accessed Online: 2023-08-17.

[2] auth0, "What is oauth 2.0?" https://auth0.com/intro-to-iam/what-is-oauth-2, 2023, accessed Online: 2023-08-17.

[3] B. Krishnamurthy, D. Malandrino, and C. E. Wills, "Measuring privacy loss and the impact of privacy protection in web browsing," in Proceedings of the 3rd Symposium on Usable Privacy and Security, 2007, pp. 52–63.

[4] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov, "Investigating users' perspectives of web single sign-on: Conceptual gaps and acceptance model," ACM Transactions on Internet Technology (TOIT), vol. 13, no. 1, pp. 1–35, 2013.

[5] R. Gafni and D. Nissim, "To social login or not login? exploring factors affecting the decision," Issues in Informing Science and Information Technology, vol. 11, no. 1, pp. 57–72, 2014.

[6] M. Urueña, A. Muñoz, and D. Larrabeiti, "Analysis of privacy vulnerabilities in single sign-on mechanisms for multimedia websites," Multimedia Tools and Applications, vol. 68, pp. 159–176, 2014.

[7] C. Guo, J. Lin, Q. Cai, W. Wang, F. Li, Q. Wang, J. Jing, and B. Zhao, "Uppresso: Untraceable and unlinkable privacy-preserving single sign-on services," arXiv preprint arXiv:2110.10396, 2021.

[8] E. Maler and D. Reed, "The venn of identity: Options and issues in federated identity management," IEEE security & privacy, vol. 6, no. 2, pp. 16–23, 2008.

[9] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov, "What makes users refuse web single sign-on? an empirical investigation of openid," in Proceedings of the seventh symposium on usable privacy and security, 2011, pp. 1–20.

[10] 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). IEEE, 2018, pp. 1–6.

[11] Z. Zhang, M. Król, A. Sonnino, L. Zhang, and E. Rivière, "El passo: Efficient and lightweight privacy-preserving single sign on," Proceedings on Privacy Enhancing Technologies, vol. 2021, no. 2, pp. 70–87, 2021.

[12] M. Kroschewski and A. Lehmann, "Save the implicit flow? enabling privacy-preserving rp authentication in openid connect," Proceedings on Privacy Enhancing Technologies, 2023.

[13] 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, 2023, pp. 25–28.

[14] 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 Proceedings 20. Springer, 2001, pp. 93–118.

[15] J. CAMENISCH, "Signature schemes and anonymous credentials from bilinear maps," in Advances in Cryptology-CRYPTO 2004. Springer-Verlag, 2004, pp. 56–72.

[16] C. Garman, M. Green, and I. Miers, "Decentralized anonymous credentials," Cryptology ePrint Archive, 2013.

[17] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong diffie hellman assumption revisited," in Trust and Trustworthy Computing: 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings 9. Springer, 2016, pp. 1–20.

[18] C. Paquin and G. Zaverucha, "U-prove cryptographic specification v1. 1," Technical Report, Microsoft Corporation, 2011.

[19] R. Xu, S. Yang, F. Zhang, and Z. Fang, "Miso: Legacy-compatible privacy-preserving single sign-on using trusted execution environments," in 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P). IEEE, 2023, pp. 352–372.

[20] W3C, "Verifiable credentials data model v2.0," https://www.w3.org/TR/vc-data-model-2.0/, 2024, accessed 9/2/2024.

[21] Auth0, "Proof key for code exchange (pkce) in web applications with spring security," https://auth0.com/blog/pkce-in-web-applications-with-spring-security/, 2024.

[22] Okta, "Implement authorization by grant type," https://developer.okta.com/docs/guides/implement-grant-type/clientcreds/main/, 2024.

[23] OAuth, "Oauth 2.0 implicit grant," https://oauth.net/2/grant-types/implicit/, 2024.

[24] Okta, "The best practice around implicit in oauth 2.0 is changing," https://developer.okta.com/blog/2019/05/01/is-the-oauth-implicit-flow-dead, 2024.

[25] J. Sedlmeir, R. Smethurst, A. Rieger, and G. Fridgen, "Digital identities and verifiable credentials," Business & Information Systems Engineering, vol. 63, no. 5, pp. 603–613, 2021.

[26] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in Proceedings of the 2014 acm sigsac conference on computer and communications security, 2014, pp. 1205–1216.

[27] J. Camenisch, M. Drijvers, P. Dzurenda, and J. Hajny, "Fast keyed-verification anonymous credentials on standard smart cards," in ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34. Springer, 2019, pp. 286–298.

[28] O. Goldreich, Foundations of cryptography: volume 2, basic applications. Cambridge university press, 2001, vol. 2.

[29] J. Camenisch and E. Van Herreweghen, "Design and implementation of the idemix anonymous credential system," in Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pp. 21–30.

[30] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22. Springer, 2002, pp. 61–76.

[31] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in Advances in Cryptology–CRYPTO 2004, 2004.

[32] D. Boneh and X. Boyen, "Short signatures without random oracles," in International conference on the theory and applications of cryptographic techniques. Springer, 2004, pp. 56–73.

[33] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5. Springer, 2006, pp. 111–125.

[34] Okta, "Auth0 documentation," https://auth0.com/docs, accessed 9/4/2024.

[35] OneLogin, "OneLogin homepage," https://www.onelogin.com/, accessed 9/4/2024.

[36] W3C, "Verifiable credentials data model v2.0," https://www.w3.org/TR/vc-data-model-2.0/, 2023, accessed Online: 2023-08-22.

[37] I. Alom, R. M. Eshita, A. I. Harun, M. S. Ferdous, M. K. B. Shuhan, M. J. M. Chowdhury, and M. S. Rahman, "Dynamic management of identity federations using blockchain," in 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2021, pp. 1–9.

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

[39] I. E. T. F. (IETF), "The bbs signature scheme," https://identity.foundation/bbs-signature/draft-irtf-cfrg-bbs-signatures.html, 2024.

[40] J. Doerner, Y. Kondi, E. Lee, A. Shelat, and L. Tyner, "Threshold bbs+ signatures for distributed anonymous credential issuance," in 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023, pp. 773–789.

[41] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," J. Cryptology, vol. 1, pp. 77–94, 1988.

[42] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in Advances in Cryptology-CRYPTO'89 Proceedings 9. Springer, 1990, pp. 239–252.

[43] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in Conference on the theory and application of cryptographic techniques. Springer, 1986, pp. 186–194.
[44] Y. Seurin, "On the exact security of schnorr-type signatures in the random oracle model," in Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2012, pp. 554–571.
[45] H. Yu, C. Du, Y. Xiao, A. Keromytis, C. Wang, R. Gazda, Y. T. Hou, and W. Lou, "Aaka: An anti-tracking cellular authentication scheme leveraging anonymous credentials," in Network and Distributed System Security Symposium (NDSS), 2023.
[46] D. Chaum, "Blind signatures for untraceable payments," in Advances in Cryptology: Proceedings of Crypto 82. Springer, 1983, pp. 199–203.
[47] D. Schröder and D. Unruh, "Security of blind signatures revisited," Journal of Cryptology, vol. 30, pp. 470–494, 2017.
[48] J. Katz, J. Loss, and M. Rosenberg, "Boosting the security of blind signature schemes," in Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27. Springer, 2021, pp. 468–492.
[49] S. Faust, C. Hazay, D. Kretzler, L. Rometsch, and B. Schlosser, "Non-interactive threshold bbs+ from pseudorandom correlations," Cryptology ePrint Archive, 2023.
[50] I. Foundation, https://identity.foundation/bbs-signature/draft-blind-bbs-signatures.txt, 2024.
[51] O. C. D. C. Registration, "Openid," https://openid.net/specs/openid-connect-registration-1_0-errata2.html, 2024.
[52] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE transactions on information theory, vol. 31, no. 4, pp. 469–472, 1985.
[53] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich, "How to win the clonewars: efficient periodic n-times anonymous authentication," in Proceedings of the 13th ACM conference on Computer and communications security, 2006, pp. 201–210.
[54] Microsoft, "What is fido2?" https://www.microsoft.com/en-us/security/business/security-101/what-is-fido2, 2024.
[55] S. Bowe, "Bls12-381: New zk-snark elliptic curve construction," https://electriccoin.co/blog/new-snark-curve/, 2024.
[56] L. Harchandani, "Bbs and bbs+ signatures and protocols for proof of knowledge of signature," https://crates.io/crates/bbs_plus, 2024.
[57] 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, 2015, pp. 1358–1369.
[58] T.-W. Lin, C.-L. Hsu, T.-V. Le, C.-F. Lu, and B.-Y. Huang, "A smartcard-based user-controlled single sign-on for privacy preservation in 5g-iot telemedicine systems," Sensors, vol. 21, no. 8, p. 2880, 2021.
[59] Z. A. Lux, D. Thatmann, S. Zickau, and F. Beierle, "Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials," in 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS). IEEE, 2020, pp. 71–78.
[60] H. Yildiz, C. Ritter, L. T. Nguyen, B. Frech, M. M. Martinez, and A. Küpper, "Connecting self-sovereign identity with federated and user-centric identities via saml integration," in 2021 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2021, pp. 1–7.
[61] E. Commission, "General data protection regulation (gdpr), chapter ii, article 5," https://gdpr-info.eu/art-5-gdpr/, 2024.
[62] S. Hammann, R. Sasse, and D. Basin, "Privacy-preserving openid connect," in Proceedings of the 15th ACM Asia conference on computer and communications security, 2020, pp. 277–289.
[63] J. He, L. Lei, Y. Wang, P. Wang, and J. Jing, "Arpsso: An oidc-compatible privacy-preserving sso scheme based on rp anonymization," in European Symposium on Research in Computer Security. Springer, 2024, pp. 268–288.
[64] M. Kroschewski, A. Lehmann, and C. Özbay, "Oppid: Single sign-on with oblivious pairwise pseudonyms," Cryptology ePrint Archive, 2024.
[65] C. Baum, T. Frederiksen, J. Hesse, A. Lehmann, and A. Yanai, "Pesto: proactively secure distributed single sign-on, or how to trust a hacked server," in 2020 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2020, pp. 587–606.
[66] S. Agrawal, P. Miao, P. Mohassel, and P. Mukherjee, "Pasta: password-based threshold authentication," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 2042–2059.
[67] S. Jarecki, A. Kiayias, and H. Krawczyk, "Round-optimal password-protected secret sharing and t-pake in the password-only model," in Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20. Springer, 2014, pp. 233–253.
[68] NIST, https://pages.nist.gov/800-63-4/sp800-63c.html, 2024.
[69] O. Foundation, "What is openid for verifiable credentials," https://openid.net/sg/openid4vc/specifications/, 2024.
[70] K. N. Chadwick and J. Vercammen, "OpenID for verifiable credentials," https://openid.net/wordpress-content/uploads/2022/05/OIDF-Whitepaper_OpenID-for-Verifiable-Credentials_FINAL_2022-05-12.pdf, 2022.
[71] P. S. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in International workshop on selected areas in cryptography. Springer, 2005, pp. 319–331.
[72] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," Journal of cryptology, vol. 21, no. 2, pp. 149–177, 2008.
[73] C. Mainka, V. Mladenov, J. Schwenk, and T. Wich, "Sok: single sign-on security—an evaluation of openid connect," in 2017 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2017, pp. 251–266.
[74] Y. Zhou and D. Evans, "Ssoscan: Automated testing of web applications for single {Sign-On} vulnerabilities," in 23rd USENIX Security Symposium (USENIX Security 14), 2014, pp. 495–510.
[75] S. G. Morkonda, S. Chiasson, and P. C. van Oorschot, "Empirical analysis and privacy implications in oauth-based single sign-on systems," in Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society, 2021, pp. 195–208.
[76] R. Wang, S. Chen, and X. Wang, "Signing me onto your accounts through facebook and google: A traffic-guided security study of commercially deployed single-sign-on web services," in 2012 IEEE Symposium on Security and Privacy. IEEE, 2012, pp. 365–379.
[77] S.-T. Sun and K. Beznosov, "The devil is in the (implementation) details: an empirical analysis of oauth sso systems," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 378–390.

Appendix

A. BBS Signature

1) Bilinear Maps: Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups of prime order p. A map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ must satisfy the following properties:

- **Bilinearity**: $e_b(g_1^x, g_2^y) = e_b(g_1, g_2)^{xy}$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, and $x, y \in \mathbb{Z}_p$.
- **Non-degeneracy**: For all generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $e_b(g_1, g_2) \neq 1$, generates $\mathbb{G}_T$.
- **Efficiency**: There exists an efficient algorithm to compute $e_b(a, b)$ for any $a \in \mathbb{G}_1$, $b \in \mathbb{G}_2$.

2) Type-3 Paring: Type-3 pairings allow for the most efficient and secure operations in $\mathbb{G}_1$ using BN curves [71]. We describe our scheme in a type-3 setting, i.e., assuming $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism $\mathbb{G}_2 \to \mathbb{G}_1$ exists.

B. The Signature Scheme

In a typical setup, BBS signature builds on pairing-based cryptography. Next, we describe a non-interactive version of the BBS signature protocol from [38], [39] that is adopted by our system. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be bilinear groups of prime order p, g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. More details on the bilinearity and pairing assumptions are provided in Appendix A.

$\underline{BBS.KeyGen:}$

- Randomly sample a vector of group elements $(h_1, \ldots, h_L) \leftarrow \mathbb{G}_1^L$.
- Uniformly choose a private key $x \leftarrow \mathbb{Z}_p^*$.
- Compute $w = g_2^x$, where g_2 is a generator of the group $\mathbb{G}_2$.
- Set **Secret Key** $sk = x \in \mathbb{Z}_p^*$
- Set **Public Key** $pk = (w, h_1, \ldots, h_L) \in \mathbb{G}_2 \times \mathbb{G}_1^L$:

$\underline{BBS.Sign:}$

- Inputs: Message $(m_1, m_2, \ldots, m_L) \in \mathbb{Z}_p^L$, and the Secret key $x \in \mathbb{Z}_p^*$.
- Select a random nonce, $e \leftarrow \mathbb{Z}_p$.
- Compute $B = g_1 \prod_{i=1}^{L} h_i^{m_i}$ and $A = B^{\frac{1}{e+x}}$.
- The **signature** σ is the tuple $(A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$.

$\underline{BBS.VerifySign:}$

- Inputs: Public key $(w, h_1, \ldots, h_L) \in \mathbb{G}_2 \times \mathbb{G}_1^L$, Message $(m_1, m_2, \ldots, m_L) \in \mathbb{Z}_p^L$, and signature $(A, e) \in \mathbb{G}_1 \times \mathbb{Z}_p$.
- Compute the following pairings and check the equality, $e_b(A, wg_2^e) \overset{?}{=} e_b(B, g_2)$, where e_b is the bilinear pairing function.
- Accept the signature if the equality holds, reject otherwise.

C. Security Assumptions

1) q-Strong Diffie-Hellman (q-SDH) Assumption: The q-SDH assumption, as introduced by Boneh and Boyen for type-3 pairing settings [72], can be stated as follows [17]:

The q-SDH assumption states that, given the tuple

$$(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2,$$

where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p with generators g_1 and g_2, and $x \in \mathbb{Z}_p$ is randomly chosen and unknown, it is computationally infeasible for any probabilistic polynomial-time adversary to produce a pair

$$(c, g_1^{1/(x+c)}) \in \mathbb{Z}_p \setminus \{-x\} \times \mathbb{G}_1.$$

Here, $c \neq -x$ ensures that $1/(x+c)$ is well-defined.

2) Decisional Diffie-Hellman (DDH) Assumption: Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. Given the tuple

$$(g, g^a, g^b, g^z) \in \mathbb{G}^4,$$

where $a, b \in \mathbb{Z}_p$ are chosen uniformly at random, the goal is to determine whether $z = ab$ or $z \in \mathbb{Z}_p$ is chosen uniformly at random.

The DDH assumption asserts that it is computationally infeasible for a probabilistic polynomial-time adversary to distinguish the tuple (g, g^a, g^b, g^{ab}) from a random tuple (g, g^a, g^b, g^z).

3) Strong Computational Diffie-Hellman Inversion (SCDHI) Problem: The Strong Computational Diffie-Hellman Inversion (SCDHI) problem, as defined in [27], [45], is a variant of the classical Computational Diffie-Hellman (CDH) problem with an additional inversion requirement. Specifically, given g^x and g^y for unknown $x, y \in \mathbb{Z}_p$, where g is the generator of a cyclic group $\mathbb{G}$, the goal is to compute:

$$g^{1/(x-y)},$$

where $x \neq y$.

The SCDHI problem is considered computationally hard because it combines the CDH challenge of dealing with unknown exponents x and y with the additional requirement of computing the inverse of the difference $x - y$.

D. More Related Work on SSO Security and Privacy Issues

SSO is a fundamental service of a typical Identity Management (IM) framework generally implemented by the IdP and serves the users and RPs. Popular IM standards such as OIDC and SAML (Security Assertion Markup Language) designs require IdP to take the crucial role of managing user identity databases and authentication servers. With so much data and services routing from a single site, IdP becomes an attractive target for data theft and interruption [8], [9]. Even if IdP is assumed to follow security policies, its availability and integrity directly affect user data privacy, its subscriber RPs, and their businesses. Moreover, studies reveal OIDC implementation flaws resulting in IdP confusion and malicious endpoints [73]. Eyeing the potential exposure of SSO authentication tokens, Zhou et al. [74] propose a framework, SSOScan, for vulnerability testing by simulating attacks and monitoring traffic. In SPRESSO [57], the authors propose a similar tool and use different types of attacker models to analyze security issues.

Apart from the security and availability aspects of SSO, user privacy and tracking is an active area of research in this domain. Researchers have found sensitive information compromised from the URL parameters and HTTP Referrer header and SSO providers like Facebook Connect allowing websites to log user actions [6]. The empirical analysis presented by [75] highlights that at least one category of personal data stored and managed by major IdP such as Google, Facebook, Apple, and LinkedIn is privacy-intrusive. Moreover, the study found privacy-friendly login options listed towards the end. [76] also analyzed the major SSO and IdPs and reported several vulnerabilities including modification of identity and unauthorized access.

On the other hand, RPs have also been violating privacy guidelines. Starting from implementation flaws to colluding. [75], [77] discuss the improper implementation and design in the architecture concerning storage, relaying, and validation of authentication tokens can also expose user credentials and enable unauthorized access. Moreover, RPs may combine different authentication tokens to generate a user profile, creating heightened risks of linkage attack [6]–[8]. In summary, a concerned user has very limited scope for auditing the consequences of the personal information stored and exchanged between the IdP and RPs.