Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière

EL PASSO: Efficient and Lightweight Privacy-preserving Single Sign On

Abstract: Anonymous credentials are a solid foundation for privacy-preserving Single Sign-On (SSO). They enable unlinkable authentication across domains and allow users to prove their identity without revealing more than necessary. Unfortunately, anonymous credentials schemes remain difficult to use and complex to deploy. They require installation and use of complex software at the user side, suffer from poor performance, and do not support security features that are now common, such as two-factor authentication, secret recovery, or support for multiple devices. In contrast, Open ID Connect (OIDC), the *de facto* standard for SSO is widely deployed and used despite its lack of concern for users' privacy. We present EL PASSO, a privacy-preserving SSO system based on anonymous credentials that does not trade security for usability, and can be incrementally deployed at scale alongside Open ID Connect with no significant changes to end-user operations. EL PASSO client-side operations leverage a WebAssembly module that can be downloaded on the fly and cached by users' browsers, requiring no prior software installation or specific hardware. We develop automated procedures for managing cryptographic material, supporting multi-device support, secret recovery, and privacy-preserving two-factor authentication using only the built-in features of common Web browsers. Our implementation using PS Signatures achieves 39x to 180x lower computational cost than previous anonymous credentials schemes, similar or lower sign-on latency than Open ID Connect and is amenable for use on mobile devices.

Keywords: Single Sign-On, Privacy

Zhiyi Zhang: UCLA, CA, USA. E-mail: zhiyi@cs.ucla.edu
Michał Król: City, University of London, United Kingdom. E-mail: michal.krol@city.ac.uk
Alberto Sonnino: Facebook Calibra and University College London, United Kingdom. E-mail: alberto.sonnino@ucl.ac.uk
Lixia Zhang: UCLA, CA, USA. E-mail: lixia@cs.ucla.edu
Etienne Rivière: ICTEAM, UCLouvain, Belgium. E-mail: etienne.riviere@uclouvain.be

1 Introduction

Single Sign-On (SSO) is an answer to the complexity and fragility of using individual passwords on the web, *i.e.*, leading to reuse and leaks [1]. SSO enables the use of a unique identity provided by an Identity Provider (IdP). Users authenticate themselves to services (called Relying Parties–RP) with tokens provided by their IdP. SSO improves overall web security [2] and enables the generalization of good security practices such as the use of 2-factor authentication (2FA) [3].

Limitations of OpenID Connect. OpenID Connect (OIDC) is a dominant SSO solution used by over a million websites in 2020 [4]. Major web players such as Google or Facebook play the role of IdPs, offering so-called *social login* features to RPs previously registered with their services. However, while facilitating identity management, the wide adoption of OIDC raises concerns on users' privacy [5, 6]. These concerns are direct consequences of the synchronous and coupled mode of operation of OIDC, illustrated in Figure 1. Each login request to an RP requires first an interaction between the user and the IdP for authentication and then another interaction between the RP and the IdP to validate credentials. An IdP is, therefore, aware of its users' every sign-on attempt and can infer private information from the nature of visited websites [7]. Similarly, RPs learn users' identifiers at IdPs systematically, enabling to aggregate identity elements shared to different RPs and violate *purpose limitation* (*e.g.*, as defined in the EU GDPR [8]). OIDC extensions have been proposed to
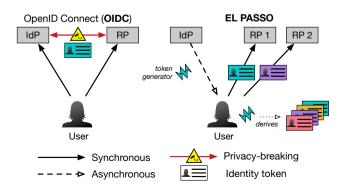

Fig. 1. Sign-on in OIDC and in EL PASSO.

increase users' privacy but only partially address these problems, as they either still leak users' global identifiers to the RPs [9] or do not prevent the IdP from logging visited websites [10]. We note, finally, that in addition to privacy concerns, the synchronicity in OIDC impacts *availability*: Users simply cannot connect to an RP if their IdP is offline. This requirement of high availability can prevent small organizations (*e.g.*, digital rights NGOs) from offering an alternative to tech giants' IdPs and counter Internet consolidation [11].

State of the Art: Anonymous Credentials. Preserving privacy in SSO requires to decouple the generation of authentication material by the IdP from its use at some RP (*i.e.*, to make it *asynchronous*), and to guarantee that the use of authentication material by the same user across RPs cannot be correlated.

Anonymous credentials [12–16] have been identified in previous work as the foundational building block for enabling privacy-preserving SSO. Anonymous credentials can be generated ahead of authentication time and stored by the user, allowing to *decouple* the generation of authentication material by an IdP and its use at some RP, *i.e.*, to make it *asynchronous*. Authentication by the same user to different RPs is unlinkable, preventing correlation of identity information shared with different RPs.

Several authentication schemes already build on anonymous credentials [17–20]. These works suffer, however, from a number of limitations that prevent their adoption as drop-in replacements of OIDC for privacy-preserving SSO at a large scale. First, they suffer from poor performance [12, 17] or overheads increasing with the number of unlinkable uses and therefore with the number of RPs [14, 15]. Second, they require users to pre-install specific software and manage manually cryptographic material [18, 21, 22]. The tasks are too complicated for most Web users [23, 24], hindering the deployment of those systems at a large scale. Finally, they lack support for features that are now considered compulsory by many clients and websites relying on SSO: they do not easily allow the user to use multiple devices (*e.g.*, their laptop and their smartphone) or are vulnerable in case of the theft of one of these devices, and they do not support two-factor authentication (2FA), *i.e.*, the possibility for an RP to require a joint sign-on operation by the same user but from two different devices.

Objective. Our objective in this paper is to bridge the gap between the simplicity, ease of deployment and practicality of OIDC, and privacy-preserving SSO based on anonymous credentials. We target a complete and integrated system for privacy-preserving SSO that does not compromise performance or ease of use in favor of security.

Contributions. We present the design, security analysis, and evaluation of EL PASSO, a practical privacy-preserving SSO system, illustrated in Figure 1.

EL PASSO is asynchronous and offers unlinkable authentication and the strong privacy guarantees of anonymous credentials. The generation of authentication material by the IdP is decoupled *in time* from its use by the client to sign on at some RP. It enables minimal disclosure of information: Users may share only elements necessary for a specific RP or provide authenticatable personal properties to an RP, such as being above a minimum age or coming from a certain geographical area, without sharing their exact age or location. The design of EL PASSO acknowledges the practical consideration that *unbreakable* anonymity is not desirable for many online services. EL PASSO offers guardrails to the risk of digital impunity associated with minimal disclosure of information, by providing optional support for *accountability* guarantees to RPs about users signing on their services. Users convicted of fraudulent behaviors (*e.g.*, authors of hate speech or harassment in an online forum, or publishers of illegal content) can be eventually identified. This identification obeys a strict cooperation process involving *several* authorities, whose number and identity must be announced *a priori* by RPs using the feature.

EL PASSO aims for ease of deployment by RPs and IdPs, and simplicity for its users. User-side operations are supported by a client module in WebAssembly [25] received on the fly from the IdP and cached in the user's browser. The module automatically manages cryptographic material, storing it using the browser built-in password manager. As a result, our platform does not require prior software installation or specific hardware and exposes user interactions similar to OIDC.

EL PASSO provides support for features that are typically expected from non-privacy-preserving SSO. We develop a double-secret scheme to enable IdP-backed, multi-device deployments. EL PASSO is robust against the theft or loss of a device and the secrets it contains and exposes simple recovery procedures at the IdP. The usage of device-specific secrets naturally supports 2FA without disclosing the user's phone or email address. In addition, we provide guidelines for incremental deployment and show how EL PASSO can operate alongside OIDC.

EL PASSO is built using PS signatures [26, 27] and designed to limit the amount of heavy cryptographic operations required for all parties. Our evaluation using representative user devices and RP and IdP services hosted on Amazon EC2 indicates that EL PASSO performance, costs, and scalability make it amenable for large-scale deployments. Sign-on operations only require one round-trip between the user-side client and the RP, and while more computations are required at the user side than for OIDC, their CPU cost is a factor of 39x to 180x lower than for those of IRMA [12, 17], a previous scheme using anonymous credentials. This results in comparable or even lower sign-on latency compared to OIDC, *e.g.*, only 250 ms on a laptop and 800 ms on a Raspberry Pi representative of a mobile device. Finally, implementations of the RP and of the IdP scale vertically and horizontally in the cloud, and allow throughput of more than 260 setup phases or more than 170 sign-on phases per second using only a 4-core VM.

Outline. We first refine our model and design goals in Section 2. We provide an overview of the design of EL PASSO in Section 3. We present its detailed construction, starting with background on anonymous credentials and zero-knowledge proofs in Section 4, followed by the protocol in Section 5 and its implementation in Section 6. We provide a security analysis in Section 7. Our evaluation is given in Section 8. We review related work in Section 9 and conclude in Section 10.

2 Design Goals

We start by defining our system and adversary models. We follow up by specifying target properties for authentication, privacy, accountability, availability, and ease of deployment.

2.1 System and Adversary Model

Our system model aligns with that of OIDC, with three actors. Relying Parties (RPs) are interested in allowing users to sign up with their services without creating specific accounts. Users trust IdPs for safeguarding their identity and associated *attributes*, and for providing the client code implementing user-side operations. RPs can choose which Identity Providers (IdP) they trust for certifying the authenticity of users. We assume that users employ a modern web browser supporting sandboxed code execution and an integrated password manager

Authentication	
personal authentication	only legitimate IdPs users can authenticate
intra-RP linkability	prevent creation of Sybils within a domain
Privacy Protection	
selective attribute disclosure	user only discloses necessary attributes
provable personal properties	user attributes' properties attested by IdP
tracking protection	IdP not aware of user's sign-ons
inter-RP unlinkability	sign-ons across multiple RPs cannot be linked
Accountability	
reliable identity retrieval	misbehaving users identity can be revealed
Deployment	
asynchronous authentication	can sign on even if IdP temporarily unavailable
no RP registration	RP does not have to register with IdP
browser-only	no software pre-installation required
multi-device support	support device theft & 2FA

Table 1. Target properties of EL PASSO.

(*i.e.*, the ability to safeguard passwords or other secrets under the user's local credentials).

We consider the following adversarial model. IdPs are considered honest-but-curious: They do not modify the protocol or deny service. Both IdPs and RPs may wish to break privacy guarantees or obtain authentication information allowing to impersonate users at correct RPs. RPs may arbitrarily deviate from the protocol in addition to observing interactions with their users. In particular, they can provide arbitrary code to run in their users' browsers. We consider, however, that this code runs in isolation from the rest of the system, and notably from the EL PASSO client that is obtained from the IdP. Users may, finally, actively attempt to abuse or bypass authentication or accountability mechanisms. Note that the adversary can control multiple corrupted entities simultaneously, *e.g.*, set up several RPs, or a combination of users and RPs.

2.2 Target Properties for EL PASSO

EL PASSO provides the properties listed in Table 1:

Authentication. EL PASSO only allows legitimate users registered with an IdP to sign up and on with an RP. It prevents any other entity in the system from impersonating existing user accounts created at RPs (*personal authentication*). This includes IdPs, who are not allowed to possess material for signing on as a specific user at an RP, to prevent account abuse in case of a data leak.

Authentication requirements also include the prevention of Sybil identities, disallowing a user from creating multiple identities for the same domain. RPs can detect authentication attempts made with credentials issued by an IdP for the same user (*intra-RP linkability*).

Privacy Protection. EL PASSO targets *minimal disclosure of information*, *i.e.*, the ability for users to control the amount of information about their profile they wish to share with RPs. A user can select which of their attributes (*e.g.*, email address, but not last name) should be revealed to an RP. Note that a user still benefits from personal authentication when sharing *none* of their personal attributes (*selective attribute disclosure*). A user may even decide to only share authenticatable certifications of *properties* about their attributes, without disclosing their values (*provable personal properties*). For instance, the 2005 Gambling Act of the United Kingdom [28] requires users of online casinos to be at least 18 years old, and holds online services responsible to enforce the regulation. In this example, EL PASSO can provide a certificate that a specific user is over 18 years old, while their actual age does not need to be revealed.

EL PASSO prevents the tracking of users' activity. It is infeasible for IdPs to *track* the sign-ons activity of their users onto different RPs, to prevent profiling and the resulting leakage of personal information [7] (*tracking protection*). In addition, in the absence of common information, it is impossible to correlate multiple accounts created from the same credential on different RPs (*inter-RP unlinkability*). For instance, an account on one RP disclosing the real name of a user cannot be correlated with another account, for the same user but at another RP, that only revealed the user's address.

Accountability. EL PASSO enables *accountability* of users, mitigating the risks associated with anonymous identities, and enabling privacy preservation for services such as online democracy. If a user engages in reprehensible behavior such as publishing illegal content or harassment, a set of authorities can eventually collaborate and hold them accountable, in cooperation with the IdP (*reliable identity retrieval*). RPs must announce the use of accountability, the set of authorities, and the threshold number of authorities strictly necessary for re-identification. RPs can validate that their users provide the necessary identity recovery material upon sign-on.

Deployment. SSO services become a critical part of many information systems [29]. Even large, highly redundant systems may experience downtimes, as exemplified by the recent 14-hour disruption of Facebook's services in March 2019 [30] or the Amazon AWS outage in 2018 [31]. In EL PASSO, a user does not need to be authenticated by the IdP *each time* they sign on with an RP; instead, users acquire their credentials periodically and can connect to RPs even when the IdP is temporarily offline (*asynchronous authentication*).

RPs do not need to register with IdPs to be able to trust authentication information, and it is impossible for IdPs to impersonate each other. The sign-on process is universal: RPs do not need specialized operations for a specific IdP (*no RP registration*). This improves system automation and mitigates Internet consolidation [11], as RPs becomes more independent and new, smaller IdPs can enter the market more easily.

On the user side, EL PASSO does *not* require specific hardware (*e.g.*, a trusted execution environment), physical device (*e.g.*, an external fingerprint reader or a smart card) or extra network services to offer its functionalities. It does not require, either, the installation of a specific software client, and all user-side code runs as sandboxed code inside their web browser (*browser-only*).

Finally, EL PASSO supports multi-device scenarios. It enables users to easily register new devices (*e.g.*, laptop, phone, tablet) and supports easy identity recovery in case of the theft of one device. It natively supports 2FA: An RP may request and assess that users connect from two different devices in order to sign on their services (*multi-device support*).

3 Overview

EL PASSO is the first system that supports *simultaneously* all the features listed in the previous section (as we show in our detailed analysis of related work in Section 9). Some key properties (*asynchronous authentication*, *intra-RP linkability*, *inter-RP unlinkability*) stem from the integration and use of anonymous credentials at the hearth of the EL PASSO protocol. Other properties are enabled through additional application of cryptographic constructs such as zero-knowledge proofs or El-Gamal encryptions (*provable personal properties*, *reliable identity retrieval*) and through novel protocols (*multi-device support*). Finally, our work achieves efficiency and ease of deployment by combining the use of WebAssembly and careful system design using browser built-in features (*browser only*). In the following, we
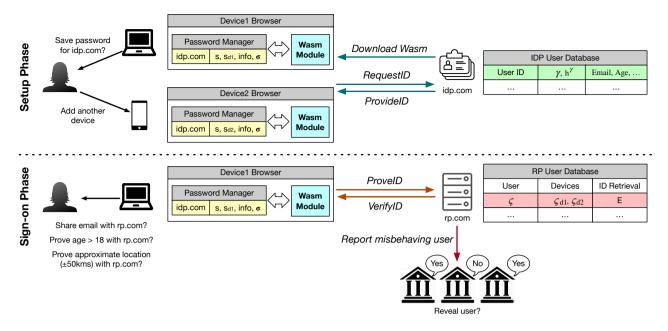
Fig. 2. An overview of EL PASSO.

provide an overview of EL PASSO operation and constituents.

EL PASSO operations are divided into two asynchronous phases (Figure 2). In the *setup phase*, the client obtains an anonymous credential from the user's IdP. In the *sign-on phase*, the client prepares an RP-specific derivation of this credential based on what information the user decides to disclose, and proves the authenticity of this client to the RP. The setup phase is executed periodically (*e.g.*, once every few days), while the sign-on phase is executed each time the user logs in or creates an account on an RP.

All user-side operations (*i.e.*, cryptographic operations) are automatically handled by a WebAssembly (Wasm) module. Wasm is an open standard that defines a portable binary-code format for executable programs that can run natively in all major web browsers. The module is referenced in the HTML pages provided by RPs and IdPs and automatically downloaded, run and later cached by the browser without user interaction. If one of the participants does not support EL PASSO, our system falls back to regular OIDC interactions (Section 6).

Setup Phase. The user first generates its secret s used later for authentication. The user authenticates to their IdP and request their credentials bound to s. The credentials include user attributes (i.e., name, age) verified by the IdP and a long-term pseudonym γ. Importantly, the credentials can be used only when possessing the user secret s, so an IdP (or anyone else) is unable to

connect to RPs on behalf of the user. Both the secret s and the obtained credentials are automatically serialized by the Wasm module and stored in the browser built-in password manager under the DNS domain of the IdP. The user does not have to manage cryptographic material manually.

Sign-on Phase. The user connects to the RP's login page that lists the information requested for login (under the principle of purpose limitation [8], requested information should be only what is strictly necessary to provide the service). The Wasm module intercepts the request and asks the user for their consent to share the requested information, using a popup.

If approved, (i) the module locally randomizes the credentials so that even if an attacker observes both the RP and the IdP it cannot link the credentials to a specific user of this IdP. (ii) The module then provides the RP with the randomized credentials and selectively discloses any subset of information embedded in the credential, enabling *selective attribute disclosure*. The client may also generate and include proofs about *properties* of the attributes they do not wish to share in the clear, enabling *provable personal properties*. The secret s is never shown to the RP; instead, the client proves knowledge of it through a zero-knowledge proof (which unconditionally hides s). Note that the client cannot generate fake attributes without forging credentials, which would require breaking the underlying PS signature scheme. (iii) The client locally generates a RP-specific pseudonym ζ. It is impossible to generate multi-

ple pseudonyms for the same RP DNS domain providing *intra-domain linkability*. Once the RP verifies the credential along with the pseudonym, it considers the user as authenticated. The credentials and the user secret required for the sign-on are automatically provided by the browser password manager.

Finally, if support for *reliable identity retrieval* is required by the RP, the client provides and proves the correctness of an El-Gamal encryption E of their long-term pseudonym γ encrypted under the public key of specific decryption authorities. If the user misbehaves, the RP discloses E to these decryption authorities, which decrypt it and collaborate with the IdP to recover the identity of the user. EL PASSO supports flexible key management for decryption authorities—the ciphertext is typically encrypted using *threshold encryption*, where at least a threshold number of decryption authorities are needed to recover the user's identifier.

Multi-device Support. A user may add a new device and use it to connect to RP accounts created with any of their older devices. Users first connect to their IdP with the new device using the standard login procedure (*e.g.*, username/password) so that the IdP can associate the new device and the old ones. Then, the new device needs to receive the secret s without leaking it to any third party; this is achieved as follows. The new device generates an ephemeral public/private key pair and sends the public key to the IdP. The user confirms the new device at the IdP using one of the older devices, and the Wasm module encrypts s under the new device's public key. The IdP sends the encrypted secret to the new device allowing it to request credentials over s. The process does not require direct communication between user devices and can be performed asynchronously. The users confirms the new device with a single button at the IdP without being exposed to the key exchange.

Two-factor Authentication. EL PASSO allows RPs to require two-factor authentication (2FA), *i.e.*, that users connect from two different devices for attesting their authenticity. Under the principle of minimal disclosure of information, 2FA does not require revealing an email address or phone number, but only to use two different previously-enabled devices. This requires, in addition to secret s, a device-specific secret s_d. The client includes s_d during the setup phase making it a part of the credentials. When the user connects with a given device to the RP for the first time, they provide ζ and generate an RP- and device-specific pseudonym ζ_d derived from s_d and the RP's DNS domain name. Similarly to ζ, ζ_d are unlinkable across domains and cannot be re-used by a malicious RP. The RP is able to link the new device to the user account using ζ and adds ζ_d to the list of authorized devices. Subsequent logins using the same device requires only providing ζ_d and do not involve additional overhead. When requiring 2FA, an RP simply checks that two subsequent logins are performed from devices with different values of s_d.

Device Theft Recovery. A user can declare the loss of a device to their IdP. The IdP will stop issuing credentials for that device. A thief able to unlock the secret storage of the stolen device's browsers would be able to connect to RPs, unless 2FA is required, but only until the IdP credential expires. It will not be able to authorize new devices. Users do not lose access to their RP accounts as long as they hold at least one device (or two devices, if 2FA is required). A user can replace their secret s using the following procedure. The user contacts the IdP and asks for credentials on a new, blinded s'; from now on, the IdP will not renew credentials for s to preserve sybil resistance. The client connects to the RP and presents credentials over the old, expired s, $\zeta(s)$, and credentials over the new s' and $\zeta(s')$. The RP replaces $\zeta(s)$ by $\zeta(s')$, and stops accepting credentials on s. The user does not need to replace the secret at all the RPs immediately, as the attacker is not able to use expired credentials.

4 Building Blocks

We present our building blocks, anonymous credentials and zero-knowledge proofs, and our cryptographic assumptions.

4.1 Anonymous Credentials

Anonymous credentials [26, 32] allow the issuance of credentials to users, and the subsequent unlinkable revelation to a verifier. Users can selectively disclose some of the attributes embedded in the credential or specific functions of these attributes. EL PASSO requires a credential scheme providing short and computationally efficient credentials, re-randomization, unlinkable multishow selective disclosure, and blind issuance [32]. An anonymous credential scheme can be defined by the set of algorithms below.

❖ **Cred.Setup(1^λ) → (pp):** define the system parameters pp with respect to the security parameter λ. These parameters are publicly available.

❖ **Cred.KeyGen(pp) → (sk, pk):** run by the authority to generate their own secret key sk and public key pk from the public parameters pp.

❖ **Cred.Issue(sk, M_h, M_p, ϕ) → (σ):** interactive protocol between the user-side client and the authority; the client obtains a credential σ embedding the set of public attributes M_p and the set of hidden attributes M_h if they satisfy the statement ϕ. Cred.Issue is composed of three algorithms:

  ❖ Cred.PrepareBlindSign(pk, M_h, ϕ) → (d, Λ, ϕ): run by the client to generate the blind factor d, and the cryptographic material Λ (embedding M_h) over which the authority blindly issues a credential.

  ❖ Cred.Sign(sk, M_p, Λ, ϕ) → ($\tilde{\sigma}$): run by the authority to issue the blinded credentials $\tilde{\sigma}$ over M_p and Λ, using their private key sk.

  ❖ Cred.Unblind($d, \tilde{\sigma}$) → (σ): run by the client to unblind $\tilde{\sigma}$ (using the factor d) to retrieve the credential σ.

❖ **Cred.Prove($pk, M_p, M_h, \sigma, \phi'$) → ($M_p, \Theta, \phi'$):** run by the client to compute a proof Θ proving possession of a credential σ certifying that the private attributes M_h and the public attributes M_p satisfy the statement ϕ'[1].

❖ **Cred.Verify(pk, M_p, Θ, ϕ') → (b):** run by any third party verifier to verify that the credential represented by the cryptographic material Θ embeds M_p as well as hidden attributes satisfying the statement ϕ', using the public key pk of the issuing authority.

All algorithms receive the security parameter λ as an input but we show it explicitly only for Cred.Setup. EL PASSO uses PS Signatures [26, 27] as the underlying credentials scheme as it uses short, and computationally efficient credentials. We use PS Signatures for the generation of credentials by the IdP, and for the verification of credentials by RPs on both known messages (*e.g.*, timestamp tp) and hidden messages (*e.g.*, user's secret s).

4.2 Zero-knowledge Proofs

Zero-knowledge proofs are protocols allowing a *prover* to convince a *verifier* that it knows a secret value x, without revealing any information about that value. The prover can also convince the verifier that they know a secret value x satisfying some statements ϕ. Anonymous credentials extensively employ zero-knowledge proofs to provide users with certified secret values; users are successively able to prove to third party verifiers that they hold secret values certified by specific credentials issuers, and prove statements about those values without disclosing them. This enables, for instance, the property of *provable personal properties*. A credential issuer may provide a user with a secret value $x = 20$ representing their age; the user can then prove in zero-knowledge to a verifier that a specific credential issuer certified that their age is larger than 18, without revealing their real age x.

EL PASSO uses non-interactive zero-knowledge proofs (NIZK) to assert knowledge and relations over discrete logarithm values. These proofs can be efficiently implemented without trusted setups using sigma protocols [33], which can be made non-interactive using the Fiat-Shamir heuristic [34] in the random oracle model.

4.3 Cryptographic Assumptions

EL PASSO inherits the same cryptographic assumptions as PS Signatures, which requires groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and satisfying (i) *Bilinearity*, (ii) *Non-degeneracy*, and (iii) *Efficiency*. We use type-3 pairings because of their efficiency [35], and therefore rely on the XDH assumption which implies the difficulty of the Computational co-Diffie-Hellman (co-CDH) problem in $\mathbb{G}_1$ and $\mathbb{G}_2$, and the difficulty of the Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_1$ [36]. We also rely on a cryptographically secure hash function H^*, hashing a string into an element of $\mathbb{G}_1$; *i.e.*, applying a full-domain hash function to hash strings into elements of $\mathbb{G}_1$ (such as BLS [36]).

5 EL PASSO Construction

We present the construction of EL PASSO satisfying all properties described in Section 2.2, and then discuss how to simplify it when *reliable identity retrieval* is not required or if the user wishes to sign on as guest with-

---

1 Note that ϕ' may be different from ϕ.

out establishing an identity with the RP, and how to support login with multiple devices. We discuss the implementation of the protocol steps in Section 6 and their security guarantees in Section 7. EL PASSO primitives (see Figure 2) are defined as follows:

Bootstrapping the IdP. The following algorithms are executed only once, when bootstrapping the IdP.

❖ **Setup(1^λ) → (pp):** output Cred.Setup(1^λ).
▷ Describe the publicly-available system parameters with respect to the security parameter λ.

❖ **KeyGen(pp) → (sk, pk):** output Cred.KeyGen(pp).
▷ Run by the IdP to generate their own secret key sk and public key pk from the public parameters pp.

Setup Phase. We describe the algorithms implementing the setup phase of EL PASSO; these algorithms are executed periodically, when the user requests a credential from the IdP.

❖ **RequestID(s) → (Λ):** set $M_h = s$ and $\phi = true$; run $(d, \Lambda, \bot) = $ Cred.PrepareBlindSign(M_h, ϕ); output Λ.
▷ Run by the user-side client to request a credential from the IdP, generating the cryptographic material Λ embedding the user secret s along with the proof. The blinding factor d will be kept by the client for later use.

❖ **ProvideID($sk, \gamma, info, tp, \Lambda$) → ($\tilde{\sigma}$):** set $M_p = (\gamma, tp, info)$; output $\tilde{\sigma} = $ Cred.Sign($sk, M_p, \Lambda, true$).
▷ Run by the IdP to provide the client with a blinded credential $\tilde{\sigma}$ over Λ, the user identifier γ, and some user attribute $info$; the credential has an expiration date tp, and is produced from the IdP's secret key sk.

❖ **UnblindID($d, \tilde{\sigma}$) → (σ):**
output $\sigma = $ Cred.Unblind($d, \tilde{\sigma}$).
▷ The client locally unblinds the credential $\tilde{\sigma}$ using the blinding factor d, and outputs the credential σ.

Sign-on Phase. We describe the algorithms implementing the sign-on phase; these algorithms are executed each time the users logs in or creates an account on an RP.

❖ **ProveID($pk, \sigma, \gamma, info, tp, domain, y$) → ($\Theta, \phi'$):**
split $info$ into $info_p$ and $info_h$, respectively containing the attributes to disclose and to hide from the RP. Set $M_p = (info_p, tp)$ and $M_h = (s, \gamma, info_h)$; pick a random $\epsilon \leftarrow \mathbb{F}$ and compute the El-Gamal ciphertext $E = (g^\epsilon, y^\epsilon h^\gamma)$, where g and h are generators of $\mathbb{G}_1$,

and y is the aggregated public key of the decryption authorities; compute

$$\tilde{h} \leftarrow H^*(domain) \; ; \; \zeta \leftarrow \tilde{h}^s$$

(where H^* is a hash function as defined in Section 4.3) and the statement

$$\phi' \leftarrow \{E = (g^\epsilon, y^\epsilon h^\gamma) \; \wedge \; \zeta = \tilde{h}^s \; \wedge \; f(info_h) = 1\};$$

compute $(\Theta, M_p, \phi') = $ Cred.Prove($pk, M_p, M_h, \sigma, \phi'$); output $(\zeta, \Theta, M_p, \phi', f)$
▷ Run by the client to show the RP a proof of correctness of user ID ζ and identity retrieval token E, and the ownership Θ of a credential σ whose attributes satisfy the statement ϕ'; this proof is generated from the RP's public domain $domain$, and from the parameters (pk, γ, tp, y). The subset of hidden attributes $info_h$ satisfy the function f.

❖ **VerifyID($pk, M_p, \Theta, \phi', domain, y$) → ($b$):** compute $\tilde{h} = H^*(domain)$ and use it to execute Cred.Verify(pk, Θ, ϕ'); output $b = 1$ if (i) the verification passes, (ii) the time-stamp tp is not expired, and (iii) the ζ and El-Gamal ciphertext E' are correctly formed; otherwise output $b = 0$.
▷ Run by the RP to verify that Θ is a proof of knowledge of a valid credential (issued by the IdP identified by the public key pk) whose attributes satisfy the statement ϕ', and user ID ζ and identity retrieve token E are correct; the proof is verified using $(M_p, domain, y)$.

Removing Reliable Identity Retrieval. In case support for reliable identity retrieval is not required by the RP (see Section 3), we can simplify the sign-on phase of the above scheme by omitting the ciphertext E; the statement ϕ' would then become $\phi' = \{\zeta = h^s \; \wedge \; f(info_h) = 1\}$, and the zero-knowledge proof Θ shorter by two field elements (if implemented, for instance, using Schnorr's protocol [33]).

Login as Guest. In case the user wishes to sign on as guest without establishing a permanent user identifier with the RP, and if the RP allows such guest sign-ons, we can simplify our scheme by omitting the group element ζ; the statement ϕ' would then become $\phi' = \{E = (g^\epsilon, y^\epsilon h^\gamma) \; \wedge \; f(info_h) = 1\}$, which shortens the proof Θ by one group element. As a result, the RP has no way to distinguish multiple sign-ons from the same user (this follows directly from the unlinkability properties of the underlying credentials scheme). The interaction between the user and the RP is still anonymous and accountable.

6 Implementation

The RP, IdP, and user-side client [37] are implemented in C++ using the MCL library [38]. The C++ implementation of the client is ported to JavaScript code using WebAssembly (Wasm) [39][2]. The use of C++ and Wasm allows our implementation to provide both high efficiency and the ability to be delivered as a web resource. The footprint of executables is 178 KB for the RP, 237 KB for the IdP, and 264 KB for the client, including the Wasm binary and JavaScript 'glue' code. All user-side operations (*i.e.*, cryptographic operations, secret sharing) are automatically handled by the Wasm client running in the browser, resulting in a deployment complexity that is similar to that of OIDC.

Obtaining Credentials and Client Code. IdP operations (authentication and RequestID) are accessed through a web page hosted in the IdP domain. Users connect to this page to obtain credentials, and the content of the page is then cached locally, including the javascript code and the Wasm client module. We leverage the fact that Wasm modules are fully cacheable by the browser [40] and can be marked as immutable [41]. When a RP wishes to authenticate a user, it redirects to the authentication page of their IdP (selected by this user from a list of IdPs trusted by this RP). The user is able to check that the authentication page URL is, indeed, part of the domain of their IdP. The user is then presented with an interface to select which attributes and provable properties they wish to present to this RP. In the majority of cases, the authentication page and Wasm module will be cached, use locally stored credentials, and there will be no direct interaction with the IdP, enabling the property of *asynchronous authentication*. When there is no cached version we favor continuity of experience by warning the user but allowing a synchronous redirection to happen towards the page hosted by the IdP, as done in SPRESSO [9]. This pragmatic approach enables continuity of experience at the cost of a minor risk on tracking protection, and users re-

quiring stronger privacy have the possibility to disable it and only use pre-authentication.

Credential/Secret Storage. The client module needs to store and later retrieve user secrets (global s and device-specific s_d) as well as the credentials received from the IdP. The client runs as a sandboxed Wasm module, unable to interact with the outside world (*e.g.*, the file system). We leverage instead the password manager service provided by the browser to store secrets and credentials securely. To be compatible with existing password manager services, EL PASSO credentials are serialized to a text format (*i.e.* Base64). There are not required modifications to the browser or password manager themselves. Users have to locally authenticate with their browser (*e.g.*, using a password, fingerprint, or face recognition) in order to grant the client access to this information. User do not need to be exposed to secrets s and s_d. They only need to know their local password and the password used at their IdP, as when using OIDC. The password manager only accepts get requests for the same domain that stored data initially, effectively protecting against attacks that would attempt to redirect the user to a fake authentication page or bypass user's scrutiny of this page domain name (*e.g.*, using a typosquatting attack [42]).

Anonymous Credentials. We implement EL PASSO using PS Signatures as the underlying credential system because of its short credentials and efficient verification. Our prototype is implemented over the curve BLS12-381 [43].

State Size. IdPs store their own key pair and a h^γ (32 Bytes) for each of their users; RPs store the public key of each IdP they trust, aggregated public keys y (32 Bytes) of decryption authorities they wish to use, as well as ciphertext E (64 Bytes) and group element ζ (32 Bytes) for each of their users. Since our implementation is based on PS Signatures, the size of the public key of the IdP increases linearly with the number of attributes, ranging from 466 Bytes (for 3 attributes) to 2,166 Bytes (for 20 attributes). Users store all the input parameters of ProveID, that is, their attributes (s, γ, *info*, *tp*), their credential σ (64 Bytes), the public key pk of the IdP who issued the credential, and the public key y (32 Bytes) of each of the decryption authorities (when *reliable identity retrieval* is required). All parties are aware of the public parameters (generated by Setup). In the simplest scenario where there exists one IdP, one user, and one RP, assuming there are 3 attributes and that reliable retrieval is required, the total state required at the IdP,

---

2 WebAssembly (Wasm) is an open standard that defines a portable binary-code format for executable programs, a corresponding textual assembly language, as well as interfaces for facilitating interactions between these programs and their host environment. Wasm code can run natively in all major web browsers. Various compilers allow transforming high-level language source code (*i.e.*, C++, Rust) into a binary file which runs in the same sandbox as regular JavaScript code.

the user, and the RP are 562 Bytes, 630 Bytes, and 594 Bytes, respectively.

Incremental Deployment Alongside OIDC. When logging into a RP website with OIDC, a user selects one of the IdPs that are supported by that RP (IdPs provides RPs with HTML snippets required to connect users). The RP then redirects the user to the chosen IdP including a list of requested attributes (*e.g.*, email address, phone number). The user must then give their consent to share the requested information. If the access is granted, the IdP responds to the RP with an ID token containing the signed user's identity (if using the "implicit flow") or returns an access code allowing RPs to fetch the ID token from the IdP (when using the "code flow").

An IdP deploying EL PASSO includes a reference to the Wasm module in the snippet provided to RPs. An RP deploying EL PASSO must use the implicit flow. Instead of redirecting the user to the IdP, the Wasm module reads the attributes requested in the URL returned by the RP and displays a popup asking for user permission to share the information. If granted, the module replies to the RP with the requested cryptographic material.

If any of the involved parties does not support EL PASSO, regular OIDC operations are used. We stress that our system does not require any additional components from the end-users and rely on mechanisms already supported by all major browsers. However, users can still choose not to use EL PASSO by setting their preferences at the IdP. In such a case, the Wasm module redirects towards IdP and does not provide the cryptographic material itself.

7 Security Analysis

We start by analyzing the security of EL PASSO against the properties defined in Section 2.2. We discuss next how to extend the security and the fault model. Finally, we discuss known attacks against incorrect OIDC implementations and the sensitivity of EL PASSO to similar attack vectors.

7.1 Achievement of Design Goals

We argue that EL PASSO satisfies the design goals described in Section 2.2 under the adversary described in Section 2.1.

Authentication. EL PASSO preserves authentication against malicious users. Authentication relies on the unforgeability of the underlying credential system—it is infeasible for malicious users to execute ProveID without holding a valid credential issued by an IdP. Furthermore, IdPs cannot take over accounts that users created with RPs as ProveID requires to provide the RP with a group element ζ that is uniquely derived from the unique user secret s, and that is persistent across authentications. The blind issuance and zero-knowledge properties of the underlying anonymous credentials system guarantee that the user secret attribute s is not revealed to the IdP (nor to any other party); hence an IdP or RP under the control of the adversary cannot impersonate existing users and access existing accounts at other RPs. In case one of the devices is compromised and the attacker is able to access the password manager on the victim user's device, this attacker will be able to log in to RPs, but only until the credentials expire. Similarly as for OIDC, this risk is mitigated by the use of 2FA and the ability to replace the secret, both features that EL PASSO enables.

Privacy Protection. The privacy guarantees of EL PASSO rely on the security (blind issuance and unlinkability) of the underlying credential scheme, and on the zero-knowledge property of the selected NIZK scheme. The blind issuance property of the underlying credential scheme ensures that RequestID does not leak any information about the secret attribute s to an honest-but-curious IdP; and zero-knowledge ensures that ProveID reveals to RPs no additional information about users' attributes than what is selectively disclosed by the user. To complete the argument, note that (i) revealing ζ does not leak s, and ζ changes indistinguishably for each website's domain (assuming a random oracle); and (ii) the ciphertext E hides γ (by the security of El-Gamal encryptions) under the assumptions of the underlying cryptographic primitives.

Accountability. EL PASSO guarantees reliable identity retrieval against misbehaving users. ProveID requires users to provide RPs with a ciphertext E, and prove in zero-knowledge that it is correctly formed; therefore, RPs can check that E is a valid encryption of the user's long-term identifier h^γ when the user signs in, even without decrypting it. If this user later misbehaves, the RP can report E to a subset of the decryption authorities (identified by the public key y) that can recover the user's long term identifier h^γ, and then collaborate with the IdP to recover the user's real-world identity. When executing ProveID, users can only disclose

or prove statements about attributes that are certified by the IdP (*i.e.*, they cannot add, remove, or modify attributes); this follows from the unforgeability of the underlying credential system, and enables provable personal properties.

7.2 Limitations and Stronger Adversaries

We now discuss limitations and possible extensions to the adversary model defined in Section 2.1.

Actively Malicious IdPs. EL PASSO only considers honest-but-curious IdPs; that is, actively malicious IdPs are not part of the threat model. An actively malicious IdP [44] can break authentication by self-issuing credentials and create fake identities; and can break accountability by refusing to cooperate with decryption authorities. It cannot, however, access an existing user account with an RP, as this is bound to the user secret s. Furthermore, an actively malicious IdP cannot be trusted to deliver the user-side Wasm module. We defer the protection against malicious IdPs to future work. One possibility would be to extend distributed SSO solutions [45, 46], where a set of IdPs must collectively authenticate a user and provide it with *shares* of their identity, and to rely on trusted standardization authorities for delivering the Wasm module. In a first iteration, this role could be played by organizations such as the W3C or digital rights NGOs. Eventually, the inclusion of this code in Web browsers, e.g. by the Mozilla foundation in Firefox, would be a more solid approach.

User Device Under Control of the Adversary. A malicious RP may inject code at the side of a honest user, but this code is sandboxed from the rest of the environment and in particular against the Wasm client module. The RP may, for instance, set up a phishing attempt by displaying a fake authentication page to the user and using a corrupted Wasm client module. The user may fail to notice that the URL does not match that of their IdP. We make the assumption that the password manager of the browser is secure and does not reveal secrets and credentials, as the domain names for the storage and retrieval of this information do not match. There have been examples, however, of successful attacks against password managers [47, 48]. The risk for EL PASSO is the same as for OIDC in this case. The solution, besides employing more secure designs for password managers [49], is to systematically require the use of 2FA. As for OIDC implementations, which do not require re-authenticating with the IdP using 2FA for *every* sign-on operations, the frequency of 2FA is a compromise between convenience and security, and can be adjusted with timestamp tp upon issuance of the credential. We note, however, that the asynchronous nature of EL PASSO prevents re-using some of the existing safety checks performed at the IdP in OIDC, such as checking for unusual origin locations of authentication requests. Such safety checks must, instead, be implemented at the RP side, possibly using provable personal properties. A second solution is to use secure login solutions such as the W3C Web Authentication protocol, WebAuthn. This standard requires the use of an external trusted device, the authenticator, for storing private keys and verifying users identity (*e.g.*, using biometrics or passcodes).

7.3 Sensitivity to Known Attacks on OIDC

We discuss attacks and exploits against incorrect implementations of OIDC [50], and the extent to which EL PASSO's design prevents similar attack vectors.

A first category of attacks exploits the coupling between the RP and the IdP in OIDC. IdP Mix-Up Attacks [51, 52] trick an honest RP to connect to a malicious IdP following the issuance of an access token, and repeating authorization codes from the user to this malicious IdP. EL PASSO uses a direct interaction between the user and the IdP, which is simpler to implement and reduces the potential for exploits.

A second category of attacks exploits the fact that in OIDC, a part of the communication between the IdP and an RP is relayed by the user browser using HTTP redirects. Code/Token/State Leakage [53, 54], CSRF Attacks and Third-Party Login Initiation [55] are examples of exploits on incorrect OIDC implementations that do not properly check redirects or embed sensitive information such as ID tokens on redirection URLs. 307 Redirect Attack [55] similarly exploit the improper use of HTTP redirection codes. EL PASSO only uses direct interactions between the user device and the IdP, in the setup phase, or an RP, in the sign-on phase, again reducing the risk of improper implementation and exploits. The redirection by the RP to the IdP cached authentication page and Wasm client may trick users, but we rely on the security of the browser's password manager to mitigate this risk.

8 Evaluation

We evaluate the EL PASSO prototype and answer the following research questions:

1. Are EL PASSO costs and usage latencies adequate to replace OIDC as an SSO solution? How does EL PASSO performance compare to anonymous credential schemes providing similar security guarantees?

2. Does the use of cryptographic operations at the user side impair the deployment of EL PASSO on low-power devices, such as mobile phones or tablets?

3. What is the scalability of EL PASSO when using an increasing number of attributes in users' profiles?

4. How do the implementations of the IdP and RP scale up when deployed in the cloud?

Setup. We deploy an IdP and an RP on two `m5ad.xlarge` instances on Amazon EC2 (4 virtual cores, 16 GB of RAM each), both in the same EC2 region. We use two representative user devices: A Dell Latitude 5590 laptop with an Intel Core i7-8650U CPU and 16 GB of RAM, and a Raspberry PI model 3b (RPI) with an A53 quad-core ARMv8 CPU and 1 GB of RAM. The RPI is representative of the performance of lower-end mobile devices such as phones or tablets. Both devices use Mozilla Firefox 76.0.1 to run the Wasm client. User operations (*e.g.*, entering a password) are emulated and instantaneous, to focus on the performance of the protocol. We are open sourcing our implementation, benchmarking scripts, and measurements data to enable reproducible results [37].

Comparison to OIDC and IRMA. We use as a first comparison point `pyoidc` [56], a complete Python implementation of OIDC. Note that the co-location of the RP and the IdP in the same EC2 zone also applies to the deployment of OIDC; this co-location is actually in favor of OIDC when measuring operation latencies. We evaluate `pyoidc` with its default settings where a standard ID token is included in the AuthN response and a single attribute is retrieved by the RP.

Our second comparison point is IRMA [17], an authentication system based on the Idemix anonymous credentials [12, 20]. IRMA is a state-of-the-art system in use by the Privacy by Design foundation [57]. We ported `irmago`, the implementation of IRMA for IoS/Android mobile platforms in `go`, to run on the same GNU/Linux

Fig. 3. Mean user-perceived operation latencies.

Fig. 4. Breakdown of the execution time of computational phases in EL PASSO and IRMA.
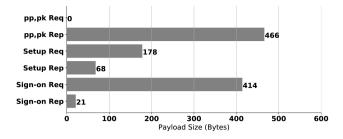
platforms as EL PASSO. We generate and deploy 4096-bit IRMA keys when issuing and verifying credentials[3].

All interactions in the three systems happen over `https`.

8.1 Latency and Costs

We start with an evaluation of the latency of operations in EL PASSO, IRMA, and OIDC. We use the laptop device, and credentials with the minimal number of 3 attributes s, γ, and tp. We analyze the impact of changing the number of attributes in a later experiment. Figure 3 presents the complete latencies as perceived by the user. These latencies include the latencies to and from the cloud, which we measured to be on average 20 ms round-trip. We present also the breakdown of computational phases in the two protocols in Figure 4, and the

---

3 While the National Institute of Standards and Technology (NIST) allows 3072-bit keys until 2030, IRMA does not support this size. 4096-bit IRMA keys have security level equivalent to our implementation of EL PASSO.

Fig. 5. Payload size of messages exchanged in EL PASSO, for credentials with 3 attributes (first two lines are for the request of the public key of the IdP by the RP).

Operation	Latency [s]	CPU time @ user [s]
EL PASSO Setup	0.72±0.16 (+190%)	0.11±0.001 (+397%)
EL PASSO Sign-on	0.82±0.18 (+125%)	0.18±0.004 (+262%)
OIDC	0.80±0.02 (+45%)	NA
IRMA Setup	30.295±0.39 (+2420%)	29.68±0.27 (+4390%)
IRMA Sign on	34.182±0.49(+2458%)	33.891±0.43 (+3640%)

Table 2. EL PASSO performance using a Raspberry PI for single and multi (M) device scenario, relative to results using a laptop from Figures 3 and 4.

size of the payload of exchanged messages in EL PASSO in Figure 5.

For EL PASSO and IRMA, we separate the asynchronous setup and sign-on phases, while sign-on in OIDC is a single, synchronous, and coupled operation. We observe that authentication in OIDC takes less time than the two EL PASSO phases combined, in part because the RP and IdP are located in the same EC2 region–In most deployments, they would be deployed in different data centers, and the RP-IdP round-trip time would add to the overall latency. However, most performance gain comes from avoiding an additional exchange between IdP and RP because the setup phase only takes place once per credential validity period, and in the majority of cases, credentials are already available at the user side. The network latency in OIDC slows the login operation (even with co-located IdP and RP) more than the more complex CPU operations introduced by EL PASSO do. IRMA experiences 5x higher Setup latency and 4x higher Sign-on latency, which is a result of much heavier cryptographic operations. Associating a new device for 2FA during the sign-on phase results in only 10ms latency increase compared to a regular sign-on.

The breakdown of computational operations in Figure 4 allows identifying the CPU time required by the different phases (note that network latencies are not shown in the breakdown). In contrast, EL PASSO requires little CPU time from the IdP, and only during the setup phase. Overall, computational costs are slightly higher for EL PASSO, but they are also more decentralized, impacting mostly users and RPs. A similar breakdown can be observed for IRMA. However, the combined execution time is 100x higher for the setup phase and 39x higher for the sign-on phase.

The amount of payload exchanged, shown in Figure 5, is reasonable. The largest payload is the sign-on request from the client to the RP and is 0.5 KB in size.

We conclude this first set of experiments with a positive answer to our two first questions: EL PASSO latencies and cost compare favorably to those of OIDC and would allow for deployment as an alternative SSO solution with negligible impact on performance or costs for users and operators of online services. Furthermore, EL PASSO significantly reduces the user-perceived latency and computational time in comparison to a similar scheme based on anonymous credentials.

8.2 Performance on Low-power Devices

As the previous experiment has shown, EL PASSO requires computation and therefore CPU time at the user side. We evaluate in this experiment whether these costs are acceptable for using it on low-power devices, such as mobile phones, tablets, or connected appliances. Our setup is the same as with the previous experiment, but using the RPI device instead of the laptop.

Table 2 compares the perceived latency using the RPI to those in Figure 3, and the total CPU time at the user side, compared to Figure 4. We can observe that the CPU cost for the setup phase almost quadruples, yet remains low at 110 ms. For the sign-on phase, the cost is multiplied by 4, primarily due to the lower performance of cryptographic operations on the ARM CPU. Yet again, the overall CPU time remains within acceptable boundaries at less than 200 ms and 220 ms when adding a new device to an account. The overall latency is impacted by both this increase in CPU time (except for OIDC), and the performance of the browser running on the RPI (including for OIDC). All operations succeed in a reasonable time, the longest being the sign-on taking a second on average, only slightly higher than OIDC compared to the previous experiment. In contrast, more complex IRMA operations experience significant execution time increase and result in Setup and Sign-on phase finishing in more than 30s. This allows us to answer positively to our second question: The performance and
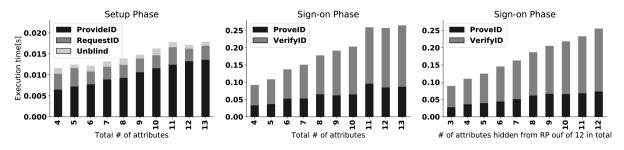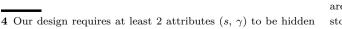
Fig. 6. Impact of the the total number of attributes and the number of attributes hidden from the RP on CPU time.

costs of EL PASSO make it adequate as a solution for SSO, even when users are equipped with low-power or mobile devices.

8.3 Scalability in the Number of Attributes

We investigate the impact of the number of attributes embedded in user credentials on the computational cost of EL PASSO. The two first plots of Figure 6 show the evolution of CPU time with a growing number of attributes, all of which are hidden from the RP. Note that the case of 3 attributes corresponds to the data in Figure 4. As expected, the CPU time increases linearly for both the setup phase and sign-on phase (first and second plot, respectively). This increase is primarily due to the additional complexity of the ProveID operation, due to the need to respectively create and validate zero-knowledge proofs for more values. Yet, the total cost, even with 13 attributes, remains reasonable, at less than a second of total CPU time. The third plot evaluates the cost of the sign-on phase when the user decides to hide an increasing number of attributes from the RP, from a profile containing 12 attributes: An abscissa value of 9 means, therefore, that the preparation of the credential for this RP only reveals 3 attributes[4]. As expected, hiding more attributes increases the computational load in the ProveID and VerifyID parts of the algorithm, yet again requiring less than a second of total CPU time. We conclude, therefore, that EL PASSO scales sufficiently well with the number of attributes to be used in practical scenarios, where the identity of a user is formed of up to a dozen different fields, answering our third question.

Fig. 7. Scaling of EL PASSO services in the cloud.

8.4 Scalability of the IdP and RP

In this last experiment, we measure the scalability of the EL PASSO implementation in the cloud, for a large number of users. We inject a growing number of precomputed requests in parallel from the laptop device and measure the achieved throughput and operation latencies. Figure 7 is a parametric plot showing the relation between the two measurements. The simpler operations required by the setup phase allow a single IdP node to handle up to 272 requests per second. The costlier sign-on phase at the RP lowers the number of operations per second to about 169[5]. These final measurements prove that EL PASSO, while involving privacy-preserving mechanisms can still be easily deployed on commodity cloud servers, and positively answer our fourth question.

9 Related Work

We review related work on SSO, its privacy-preserving extensions and anonymous authentication. We classify the most related of the systems we discuss in Table 3 us-

---

4 Our design requires at least 2 attributes (s, γ) to be hidden from RPs.

5 We note that operations at the IdP and RP for different users are naturally disjoint-access parallel, if the user information is stored in a scalable NoSQL database. This allows scaling the IdP or RP horizontally as necessary.

System	Personal Authentication	Intra-RP Linkability	Tracking Protection	Selective Disclosure	Inter-RP Unlinkability	Provable Personal Attributes	Reliable Identity Retrieval	Async. Authentication	No RP Registration	Browser-only
SAML [58]	✗	✓	○	○	✗	✗	✓	✗	✗	✓
OIDC [59]	✗	✓	○	◑	✗	✓	✓	✗	✗	✓
Apple SignIn [10]	✗	✓	○	◑	✓	✓	✓	✗	✗	✓
SPRESSO [9]	✓	✓	●	◑	✗	✓	✓	✗	✗	✓
PRIMA [60]	✓	✓	●	◑	✗	✓	✓	✗	✓	✗
UnlimitID [61]	✓	✓	◑	●	✓	✓	✗	✗	✗	✗
NextLeap [62]	✓	✓	◑	●	✓	✓	✗	✗	✓	✗
UProve [14, 15]	✓	✗	●	●	✓	✓	✗	✓	✓	✗
Privacy-ABCs [19]	✓	✓	●	●	✓	✓	✓	✓	✓	✗
IRMA [17]	✓	✗	●	●	✓	✓	✗	✓	✓	✗
Hyperledger Idemix [21]	✓	✗	●	●	✓	✓	✗	✓	✓	✗
EL PASSO	✓	✓	●	●	✓	✓	✓	✓	✓	✓

Table 3. Properties of different SSO and Anonymous Credentials systems.
Tracking Protection ◑ : UnlimitID and UnlimitID-based NextLeap rely on unlinkable credentials. However, the blinded credentials must be deposit by the users at IdP, potentially allowing IdP to perform user tracking. — **Selective Disclosure** ◑ : OIDC, Apple SignIn and SPRESSO allow to disclose a subset of user information, but are unable to prove statements about their attributes (*i.e.*, age > 18). PRIMA supports proving statement about attributes only if they are expressed as additional attributes signed by IdP.

ing the properties defined in Section 2 and summarized in Table 1.

SSO Standards. The Security Assertion Markup Language (SAML) [58] is an XML-based authentication protocol, widely deployed before OIDC was standardized. It uses a message flow that is very similar to that of OIDC, therefore shares its privacy vulnerabilities. Furthermore, SAML does not enable selective attribute disclosure and provides less flexibility to developers than OIDC.

OIDC combines the previous Open ID identity management standard with the OAuth authentication protocol [63]. The privacy issues of these protocols were pointed out as being a result of the direct IdP and RP communication [5].

SSO Extensions. Sign In with Apple [10] uses randomized per-RP identifiers (alias email addresses) for users instead of permanent identifiers (actual email address). This solution provides inter-domain unlinkability. However, Apple has largely adopted OIDC for its implementation [10, 64] and the IdP-side privacy concerns also hold true for this system.

SPRESSO [9] decouples the communication between the RP and the IdP, letting the two parties communicate indirectly through a *forwarder* agent at the client. A user sign-on request to an RP is followed by a synchronous user request to the IdP for credentials. The synchronicity of operations requires protection against time-based attacks, where the IdP could correlate requests from the user and the RP. Further-

more, SPRESSO leaks user's global ID to RPs enabling tracking

PRIMA [60] decouples communications between RP and IdP and supports selective attribute disclosure on top of Oblivion [65]. However, it requires contacting the IdP for every user sign-on and does not provide inter-RP unlinkability.

Anonymous Authentication. Anonymous credentials such as CL Signatures [13, 66] and Idemix [12, 20] are useful in personal identity management [17], anonymous attestation [67–69], and electronic cash [70]. They provide blind issuance and unlinkability through randomization, but come with significant computational overheads, and large credentials size. U-Prove [14, 15] and Anonymous Credentials Light (ACL) [71] are computationally efficient credentials that can be used once unlinkably; therefore the size of the credentials is linear with the number of unlinkable uses. Furthermore, they do not allow an RP to distinguish different sign-on attempts by the same user, and cannot provide intra-RP linkability. UnlimitID [61] builds attribute-based SSO credentials over aMAC [32], used as pseudonyms. This allows inter-RP unlinkability, as IdPs are unable to track user activity over different RPs using different pseudonyms. UnlimitID follows the main flow of OIDC and requires users to deposit their anonymized pseudonyms at IdP before RPs can access them. This may allow the IdP to correlate the deposit of a pseudonym and its request by an RP, enabling tracking. The NextLeap project [62] intends to extend Unlim-

itID [61] by storing identity and trust information in a blockchain, positioning that this would remove the need for RPs to explicitly register with IdPs. There is also no such need for RPs to register with IdPs in in EL PASSO Recent attribute-based credential [72] implementations such as IRMA [17], Privacy-ABCs [16, 19, 73] and Hyperledger Idemix [18, 21] significantly improve the performance, but still suffer from high user-perceived latency on less powerful devices. Furthermore, they require manual installation and configuration/credential management, do not enable 2FA or multi-device support. Similar issues were already identified as barriers preventing wide-spread deployment of mature security systems such as PGP [23, 24].

In combination with anonymous credentials, multiple works propose to prevent [74–77] or limit [78] login attempts of specific users without revealing their identities. While those platforms can block misbehaving users from accessing a specific RP, they are unable to hold these users accountable for their actions (*e.g.*, when publishing hate speeches online). Finally, these blacklisting systems require significant computational and communication overhead, limiting their usability and deployability, which are essential goals for EL PASSO.

10 Conclusion

We presented EL PASSO, an SSO solution that combines the security of anonymous credentials with the practicality of OIDC. Our solution protects users from being tracked by either RPs or IdPs and allows us to disclose only the minimum user information required to sign on. While providing strong privacy protection, EL PASSO can also hold misbehaving users accountable in cooperation with law enforcement authorities. Our system is implemented as a Wasm module that is downloaded on the fly and cached by the user's browser. Support for multi-device deployment, privacy-preserving 2FA, and device theft recovery is provided and only rely on the user browser's built-in features. We believe that these properties open the perspective of using our system in a wide range of use cases where the use of anonymous credentials would otherwise be an issue, such as e-democracy platforms and opinion forums.

References

[1] Blake Ives, Kenneth R Walsh, and Helmut Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4), 2004.

[2] Jason Goode. The importance of identity security. *Computer Fraud & Security*, 2012(1), 2012.

[3] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. Multi-factor authentication: A survey. *Cryptography*, 2(1):1, 2018.

[4] SimilarTech.com. Market share & web usage statistics: OpenID, 2020. Accessed: 2020-05-23.

[5] Stefan Brands. The problem(s) with OpenID, 2007. Accessed: 2020-05-23.

[6] Ruti Gafni and Dudu Nissim. To social login or not login? exploring factors affecting the decision. *Issues in Informing Science and Information Technology*, 11(1), 2014.

[7] Balachander Krishnamurthy, Delfina Malandrino, and Craig E Wills. Measuring privacy loss and the impact of privacy protection in web browsing. In *3rd symposium on Usable privacy and security*, SOUPS. ACM, 2007.

[8] European Commission. General Data Protection Regulation (GDPR), chapter II, Article 5, 2016. Accessed: 2020-05-23.

[9] Daniel Fett, Ralf Küsters, and Guido Schmitz. SPRESSO: A secure, privacy-respecting single sign-on system for the web. In *22nd Conference on Computer and Communications Security*, CCS. ACM, 2015.

[10] Apple Inc. Sign In with Apple, 2020. Accessed: 2020-05-23.

[11] Jari Arkko, Brian Trammell, Mark Nottingham, Christian Huitema, Martin Thomson, Jeff Tantsura, and Niels ten Oever. Considerations on internet consolidation and the internet architecture. Internet-Draft draft-arkko-iab-internet-consolidation-01, IETF Working Draft, March 2019.

[12] Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Tom Heydt-Benjamin, Dieter Sommer, and Greg Zaverucha. Cryptographic protocols of the identity mixer library. Technical Report RZ 3730, IBM Research – Zurich, 2009.

[13] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Annual International Cryptology Conference*. Springer, 2004.

[14] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1. *Technical Report, Microsoft Corporation*, 2011.

[15] Christian Paquin. U-prove technology overview v1.1. *Microsoft Corporation Draft Revision*, 1, 2011.

[16] Thomas Baignères, Patrik Bichsel, Robert R Enderlein, Hans Knudsen, Kasper Damgård, Jonas Jensen, Gregory Neven,

Janus Nielsen, Pascal Paillier, and Michael Stausholm. D4. 2 Final Reference Implementation. *ABC4-Trust, IBM Res., Zürich, Switzerland*, 2014.

[17] Gergely Alpár, Fabian van den Broek, Brinda Hampiholi, Bart Jacobs, Wouter Lueks, and Sietse Ringers. IRMA: practical, decentralized and privacy-friendly identity management using smartphones. In *10th Workshop on Hot Topics in Privacy Enhancing Technologies*, HotPETs, 2017.

[18] Hyperledger. MSP Implementation with Identity Mixer, 2020. Accessed: 2020-05-23.

[19] Kai Rannenberg, Jan Camenisch, and Ahmad Sabouri. Attribute-based credentials for trust. *Identity in the Information Society, Springer*, 2015.

[20] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Conference on Computer and Communications Security*, CCS. ACM, 2002.

[21] Hyperledger. Identity Mixer MSP configuration generator (idemixgen), 2020. Accessed: 2020-05-23.

[22] Privacy by Design Foundation. IRMA Mobile Client, 2020. GitHub repository. Accessed: 2020-05-23.

[23] Scott Ruoti, Nathan Kim, Ben Burgon, Timothy Van Der Horst, and Kent Seamons. Confused Johnny: when automatic encryption leads to confusion and mistakes. In *Symposium on Usable Privacy and Security*, 2013.

[24] Alma Whitten and J Doug Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*, 1999.

[25] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. Bringing the web up to speed with webassembly. In *38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI, 2017.

[26] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Cryptographers' Track at the RSA Conference*. Springer, 2016.

[27] David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In *Cryptographers' Track at the RSA Conference*, pages 319–338. Springer, 2018.

[28] United Kingdom Legislation. Gambling Act, 2005. Accessed: 2020-05-23.

[29] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey, and Konstantin Beznosov. What makes users refuse web single sign-on?: an empirical investigation of OpenID. In *7th Symposium on Usable Privacy and Security*, SOUPS. ACM, 2011.

[30] BBC News. Facebook and Instagram suffer most severe outage ever, 2019. Accessed: 2020-05-23.

[31] Jake Swearingen. When Amazon Web Services Goes Down, So Does a Lot of the Web, 2018. Accessed: 2020-05-23.

[32] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *Conference on Computer and Communications Security*, CCS. ACM, 2014.

[33] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, CRYPTO. Springer, 1989.

[34] Feng Hao. Schnorr non-interactive zero-knowledge proof. RFC 8235, RFC Editor, September 2017.

[35] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16), 2008.

[36] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology*, ASIACRYPT. Springer, 2001.

[37] Zhiyi Zhang and Michał Król. GitHub Repository of EL PASSO souce code. https://github.com/Zhiyi-Zhang/PS-Signature-and-EL-PASSO, 2020. Accessed: 2020-12-15.

[38] Mitsunari Shigeo. MCL: a portable and fast pairing-based cryptography library, 2020. GitHub Repository. Accessed: 2020-05-23.

[39] World Wide Web Consortium (W3C). Web Assembly, 2020. Accessed: 2020-05-23.

[40] MDN contributors. Caching compiled WebAssembly modules, 2020. MDN web docs. Accessed: 2020-05-23.

[41] MDN contributors. Cache-Control, 2020. MDN web docs. Accessed: 2020-05-23.

[42] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long "taile" of typosquatting domain names. In *23rd USENIX Security Symposium*, 2014.

[43] Bowe Sean. BLS12-381: New zk-SNARK Elliptic Curve Construction, 2017. Accessed: 2020-05-23.

[44] Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Do not trust me: Using malicious idps for analyzing and attacking single sign-on. In *European Symposium on Security and Privacy*, EuroS&P. IEEE, 2016.

[45] William K Josephson, Emin Gün Sirer, and Fred B Schneider. Peer-to-peer authentication with a distributed single sign-on service. In *International Workshop on Peer-to-Peer Systems*, IPTPS. Springer, 2004.

[46] Tierui Chen, Bin B Zhu, Shipeng Li, and Xueqi Cheng. Threspassport–a distributed single sign-on service. In *International Conference on Intelligent Computing*, ICICA. Springer, 2005.

[47] Zhiwei Li, Warren He, Devdatta Akhawe, and Dawn Song. The emperor's new password manager: Security analysis of web-based password managers. In *23rd USENIX Security Symposium*, 2014.

[48] David Silver, Suman Jana, Dan Boneh, Eric Chen, and Collin Jackson. Password managers: Attacks and defenses. In *23rd USENIX Security Symposium*, 2014.

[49] Daniel McCarney, David Barrera, Jeremy Clark, Sonia Chiasson, and Paul C Van Oorschot. Tapas: design, implementation, and usability evaluation of a password manager. In *28th Annual Computer Security Applications Conference*, ACSAC, 2012.

[50] Daniel Fett, Pedram Hosseyni, and Ralf Kuesters. An extensive formal security analysis of the openid financial-grade api. *arXiv:1901.11520 [cs]*, January 2019. arXiv: 1901.11520.

[51] Christian Mainka, Vladislav Mladenov, Jorg Schwenk, and Tobias Wich. Sok: Single sign-on security — an evaluation of openid connect. In *European Symposium on Security and Privacy*, EuroS&P. IEEE, 2017.

[52] Michael Jones, John Bradley, and Nat Sakimura. OAuth 2.0 mix-up mitigation. Internet-Draft draft-ietf-oauth-mix-up-mitigation-01, IETF Working Draft, July 2016.

[53] Michael Jones, Brian Campbell, John Bradley, and William Denniss. OAuth 2.0 token binding. Internet-Draft draft-ietf-

oauth-token-binding-07, IETF Working Draft, June 2018.

[54] Daniel Fett, Ralf Kuesters, and Guido Schmitz. The web sso standard openid connect: In-depth formal security analysis and security guidelines. *arXiv:1704.08539 [cs]*, April 2017. arXiv: 1704.08539.

[55] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of OAuth 2.0. In *Conference on Computer and Communications Security*, CCS. ACM, 2016.

[56] OpenIDC. pyoidc: A complete OpenID Connect implementation in Python, 2020. GitHub Repository. Accessed: 2020-05-23.

[57] Privacy By Design Foundation. Privacy by Design Foundation, 2020. Accessed: 2020-05-23.

[58] John Hughes and Eve Maler. Security assertion markup language (saml) v2.0 technical overview. Technical Report sstc-saml-tech-overview-2.0-draft-08, OASIS SSTC, 2005.

[59] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *2nd workshop on Digital identity management*. ACM, 2006.

[60] Muhammad Rizwan Asghar, Michael Backes, and Milivoj Simeonovski. PRIMA: Privacy-preserving identity and access management at internet-scale. In *International Conference on Communications*, ICC. IEEE, 2018.

[61] Marios Isaakidis, Harry Halpin, and George Danezis. UnlimitID: Privacy-preserving federated identity management using algebraic MACs. In *Workshop on Privacy in the Electronic Society*, WPES. ACM, 2016.

[62] Harry Halpin. NEXTLEAP: Decentralizing identity with privacy for secure messaging. In *12th International Conference on Availability, Reliability and Security*. ACM, 2017.

[63] Dick Hardt. The oauth 2.0 authorization framework. RFC 6749, RFC Editor, October 2012.

[64] Board of Directors of the OpenID Foundation. Open Letter from the OpenID Foundation to Apple Regarding Sign In with Apple, 2019. Accessed: 2020-05-23.

[65] Milivoj Simeonovski, Fabian Bendun, Muhammad Rizwan Asghar, Michael Backes, Ninja Marnau, and Peter Druschel. Oblivion: Mitigating privacy leaks by controlling the discoverability of online information. In *International Conference on Applied Cryptography and Network Security*, ACNS. Springer, 2015.

[66] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. Aggregating cl-signatures revisited: Extended functionality and better efficiency. In *International Conference on Financial Cryptography and Data Security*. Springer, 2013.

[67] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Conference on Computer and Communications Security*, CCS, 2004.

[68] Liqun Chen, Dan Page, and Nigel P Smart. On the design and implementation of an efficient daa scheme. In *International Conference on Smart Card Research and Advanced Applications*. Springer, 2010.

[69] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *International Journal of Information Security*, 12(3), 2013.

[70] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible e-cash made practical. In *IACR International Workshop on Public Key Cryptography*. Springer, 2015.

[71] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Conference on Computer and Communications Security*, CCS. ACM, 2013.

[72] Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. Concepts and languages for privacy-preserving attribute-based authentication. In *IFIP Working Conference on Policies and Research in Identity Management*. Springer, 2013.

[73] IBM. IBM Identity Mixer, 2015. Accessed: 2020-05-23.

[74] Patrick P Tsang, Man Ho Au, Apu Kapadia, and Sean W Smith. Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In *Conference on Computer and Communications Security*, 2007.

[75] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Conference on Computer and Communications Security*, CCS. ACM, 2006.

[76] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Australasian Conference on Information Security and Privacy*. Springer, 2007.

[77] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, EUROCRYPT. Springer, 2001.

[78] Ryan Henry and Ian Goldberg. Formalizing anonymous blacklisting systems. In *Symposium on Security and Privacy*, S & P. IEEE, 2011.