Digital Identity

with anonymous credentials
Setting

IdP

RP

User
Standard SSO

Several limitations
Standard SSO

Limitation I - Poor user and RP privacy
Standard SSO
Limitation II - Requires IdP availability
Standard SSO
Limitation III - Does not work offline
Standard SSO

Limitation III - Requires RP registration
Anonymous Credentials

This is not a new idea
Anonymous Credentials

Setup phase

- Embed many user attributes (e.g., email)
- Attributes are 'attested' by the IdP
- Can only be issued by the IdP
Anonymous Credentials

Sign-on phase

- No interaction with the IdP
- Can re-use the credential anonymously
- Can selectively show some attributes
- Can prove statements about attributes
What we get

Anon. Credentials

- Privacy
- Availability
- RP and user can be offline
- RP do not register with IdP
## What's the catch?

<table>
<thead>
<tr>
<th>Anon. Credentials</th>
<th>Standard SSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy</td>
<td>User usability</td>
</tr>
<tr>
<td>Availability</td>
<td>Performance</td>
</tr>
<tr>
<td>RP and user can be offline</td>
<td></td>
</tr>
<tr>
<td>RP do not register with IdP</td>
<td></td>
</tr>
</tbody>
</table>
EL PASSO
Privacy-preserving, Asynchronous Single Sign-On
What is it?
It is a system contribution (no new crypto)

Anon. Credentials
What is it?
It is a system contribution (no new crypto)

Anon. Credentials
with:

- User usability
- Performance

} Standard SSO
What is it?

It is a system contribution (no new crypto)

Anon. Credentials

with:

- User usability
- Performance

(Optional) Accountability

} Standard SSO
Features
User Usability

• Implemented in C++ using MCL crypto library
• User-side client ported to javascript using WebAssembly (Wasm)
Features
User Usability

• Implemented in C++ using MCL crypto library
• User-side client ported to javascript using WebAssembly (Wasm)

• Executable footprint: 178KB (including Wasm bin, js glue code)
• All user-side operations are handled by Wasm in the browser
• Wasm module cached, marked immutable, sandboxed
• User secrets stored in the browser's password manager
• User state: 600 bytes (3 attributes)
Features

Performance

Figure 3: User-perceived operation latencies.

Figure 4: Breakdown of the execution time of computational phases in EL PASSO and IRMA.

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.

Performance on low-power devices.

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.

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

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 10
Figure 3: User-perceived operation latencies.

Figure 4: Breakdown of the execution time of computational phases in EL PASSO and IRMA.

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.

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

**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 10
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.

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

### 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.

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

Decryption authorities

RP

IdP
Additional Features
More in the paper

• Multi-device support
• 2FA support
• Device theft recovery
• Login as guest
Conclusion

EL PASSO

• **Code:** https://github.com/Zhiyi-Zhang/PSSSignature
**Construction**
Anonymous credentials

**Setup Phase**

\[ \text{PrepareBlindSign}(pk, M_h, \phi) \rightarrow (d, \Lambda, \phi) \]

\[ \text{Sign}(sk, M_p, \Lambda, \phi) \rightarrow \tilde{\sigma} \]

\[ \text{Unblind}(d, \tilde{\sigma}) \rightarrow \sigma \]

**Sign-on Phase**

\[ \text{Prove}(pk, M_p, M_h, \sigma, \phi') \rightarrow (M_p, \Theta, \phi') \]

\[ \text{Verify}(pk, M_p, \Theta, \phi') \rightarrow b \]
Construction
Setup phase

RequestID(s) → Λ

Cred.PrepareBlindSign(pk, s) → (d, Λ)

ProvideID(sk, γ, info, tp, Λ) → ˜σ

Cred.BlindSign(sk, (γ, tp, info), Λ) → ˜σ

UnblindID(d, ˜σ) → σ

Cred.Unblind(d, ˜σ) → σ
ProveID(pk, σ, γ, info, tp, dns) → (ζ, Θ, Mp, φ', f')
Construction
Sign-on phase (prove Id)

ProveID\((pk, \sigma, \gamma, info, tp, dns) \rightarrow (\zeta, \Theta, M_p, \phi', f')\)

\[ \zeta = (H^*(dns))^s \]
Construction
Sign-on phase (prove Id)

\[ \text{ProvelD}(pk, \sigma, \gamma, \text{info}, \text{tp}, \text{dns}) \rightarrow (\zeta, \Theta, M_p, \phi', f') \]

\[ \text{info} \quad \text{info}_p \quad \text{info}_h \]

\[ \zeta = (H^*(\text{dns}))^s \]

\[ M_p = (\text{info}_p, \text{tp}) \]

\[ M_h = (s, \gamma, \text{info}_h) \]
### Construction

**Sign-on phase (prove Id)**

**ProveID**

\[
\text{ProveID}(pk, \sigma, \gamma, \text{info}, tp, dns) \rightarrow (\Theta, M_p, \phi'(\zeta, f))
\]

\[
\begin{align*}
\text{info} & \quad \leftrightarrow \quad \text{info}_p \quad \leftrightarrow \quad \zeta = \left(H^*(dns)\right)^s \\
& \quad \leftrightarrow \quad \text{info}_h
\end{align*}
\]

\[
M_p = (\text{info}_p, tp)
\]

\[
M_h = (s, \gamma, \text{info}_h)
\]

**Cred.Prove**

\[
\text{Cred.Prove}(pk, M_p, M_h, \sigma, \phi') \rightarrow (\Theta, M_p, \phi')
\]

\[
\phi' = \{\zeta = \left(H^*(dns)\right)^s \land f(\text{info}_h) = 1\}
Construction
Sign-on phase (verify Id)

VerifyID($pk, M_p, \Theta, dns, \phi'(<\zeta,f>)$) $\rightarrow b$

Cred.Verify($pk, \Theta, \phi(<\zeta,f>)$) $\rightarrow b'$

$b = (b' = 1 \land tp > now)$

$\zeta$ is the user id