UP-SSO: Enhancing the User Privacy of SSO by Integrating PPID and SGX

*Abstract*—Single sign-on (SSO) services are widely deployed on the Internet as the identity management and authentication infrastructure. In an SSO system, after authenticated by the identity providers (IdPs), a user is allowed to log into relying parties (RPs) by submitting an identity proof. However, SSO introduces the potential leakage of user privacy, which is indicated by NIST. That is (a) a curious IdP could track a user’s all visits to any RPs and (b) collusive RPs could link the user’s identities across different RPs, to learn the user’s activity profile. NIST suggests that the Pairwise Pseudonymous Identifier (PPID) should be adopted to prevent collusive RPs from linking the same user, as PPID mechanism enables an IdP to provide a user with multiple individual IDs for different RPs. However, PPID mechanism cannot protect users from IdP’s tracking, as it still exposes RP identity to IdP. In this paper, we propose an SSO system, named UP-SSO, providing the enhanced PPID mechanism to protect a user’s profile of RP visits from both the curious IdP and the collusive RPs by integrating PPID and SGX. It separates an IdP service into two parts, the server-side service and user-side service. The generation of PPID is shifted from IdP server to user client, so that IdP server no longer needs to learn RP ID. The integrity of user client can be verified by IdP through remote attestation. The detailed design of UP-SSO is described in this paper, and the systemic analysis is provided to guarantee its security. We implemented the prototype system of UP-SSO, and the evaluation of the prototype system shows the overhead is modest.

Keywords—SSO, PPID, Privacy, SGX

# Introduction

Single sign-on (SSO) systems, such as OAuth, OpenID Connect (OIDC) and SAML, have been widely deployed; for example, 80 websites among the Top-100 websites have integrated SSO services [1]. SSO delegates the user authentication from online web services, called *relying parties* (RPs) to a trusted third party, called the *identity provider* (IdP). With the help of SSO, a user no longer needs to maintain multiple credentials for different RPs.

However, the adoption of SSO also raises new privacy concerns. NIST [2] analyzed that the curious IdP or multiple collusive RPs could break the users’ privacy in these ways. (1) *IdP-based login tracing* - The IdP knows the identities of the RP and user in each login instance; and (2) *RP-based identity linkage* - The RP learns a user’s identity from the identity proof, so that malicious RPs could collude to link the user’s login activities.

To protect user privacy, the pairwise pseudonymous identifier (PPID) is suggested by NIST [2], OIDC [3] and SAML [4]. That is, an IdP assigns a user multiple independent identities (called PPIDs) for different RPs. Thus, an RP cannot correlate the PPID with the user’s PPID at another RP. More and more IdPs have adopted PPIDs to protect user privacy, such as Active Directory Federation Services and Oracle Access Management. NORDIC APIS and CURITY also suggest the adoption of PPIDs in SSO to protect user privacy.

Although the PPID mechanism effectively protects user from the RP-based identity linkage, the IdP-based login tracing is not well prevented in SSO systems. There have been many works to protect user privacy in SSO systems; however, to the best of our knowledge, none of the existing solutions [5]–[8] offers the comprehensive protection of user privacy. In section II-C, we will give the detailed introduction of existing solutions and explain the shortcomings of these schemes.

This paper proposes UP-SSO, the comprehensive protection against both the curious IdP and RPs. The key idea of UP-SSO is to shift the PPID generation from the IdP server to the user side, so that the IdP issues the identity proofs binding the user’s PPID (denoted as *PPIDU*), while only requiring an one-time RP pseudo-identity (denoted as *PPIDRP*). In UP-SSO system, the client takes the responsibility to generate *PPIDU* and transform the RP identity into *PPIDRP*, so the IdP-based login tracing is impossible as it no longer receives RP identities. It is worth noting that the user-side function of SSO services must be deployed at the user-controlled but IdP-trusted environment, i.e., an Intel SGX enclave. With Intel SGX, the security guarantees supported by CPU, the user-side function will be deployed at the user’s computer, and its integrity is guaranteed by *remote attestation*, the hardware-based security mechanism provided by Intel SGX.

**Our contribution.** UP-SSO provides PPID-based identity proofs but without exposing RP identities to the curious IdP, by shifting the *PPIDU* generation from the IdP server to user-controlled SGX enclaves, which are trusted by the IdP. Therefore, UP-SSO offers the comprehensive solution to hide the users’ login traces from both the curious IdP and malicious collusive RPs.

The remainder is organized as follows. Section II introduces the background and related works. We describe the threat model and design of UP-SSO in Sections III and IV, respectively. Section V analyzes the security and privacy. We present the implementation and experimental evaluation in Section VI. Section VII concludes this work.

# Background

UP-SSO is compatible with OIDC, and achieves privacy protections based on Intel SGX.

## OpenID Connect and PPID

OIDC [3] is an extension of OAuth 2.0 to support user authentication, and one of the most prominent SSO protocols.

I**mplicit Flow of User Login.** The implicit flow is the typical user login flow of OIDC. As shown in Figure 1, at the beginning, a user attempts to log into an RP (Step 1). The RP constructs a request for identity proofs, which is redirected by the user to the IdP. This request contains *IDRP* and other optional attributes (Step 2). If the user is not authenticated, the IdP initiates an authentication process (Step 3). Then, the IdP signs *IDRP* and the user’s unique identity (or PPID) into an identity proof, and sends it back to the RP (Steps 4 and 5). The RP finally verifies this identity proof, extracts the user identity, and returns the result (Steps 6 and 7).

**Pairwise Pseudonymous Identifier (PPID).** Different from the unique user identity at the IdP, a PPID is an RP-specific user identity. That is, while a user visits different RPs, the IdP will provide a different but constant user identity for each RP. NIST [2] suggests that PPIDs should be generated randomly and assigned by the IdP, or derived from a user’s information if the derivation is irreversible and unguessable.

## Intel SGX

Intel Software Guard Extensions (SGX) [9] is the hardware-based security mechanism provided by Intel processors. A task protected by SGX runs within an enclave, and this enclave can attest its run-time integrity to remote entities.

**Enclave.** The enclave’s code and data are stored in Processor Reserved Memory (PRM). PRM cannot be directly accessed by other software, including system software and System Management Module code (Ring 2). The Direct Memory Access (DMA) of PRM is also unavailable, as enclave is protected from other peripherals. That is, while the application is running inside the enclave, even the device owner cannot break its security, such as accessing or tempering the application’s data.

**Remote Attestation.** Remote attestation enables the software inside an enclave to attest to a remote entity that it is trusted. The SGX remote attestation allows a player to verify the application’s identity, intactness (never tampered), and that it is running securely within an enclave. Moreover, with the remote attestation, the secure key exchange between the player and remote enclave application is also available.

## Existing Privacy-Preserving Solutions for SSO

NIST [2] suggests that the SSO should prevent user’s trace from being tracked by both RP and IdP. Some protocols simply hide the RP’s identity from IdP, such as SPRESSO [5] (using encrypted RP identifier) and BrowserID [6] (RP identifier is added by user). An encrypted RP identity is bound in identity proofs by the SPRESSO IdP, while the BrowserID IdP firstly binds the user identity and a key pair, and the user will bind the RP identity by itself using the certified key pair. However, they are not defensive to RP-based identity linkage, and PPID cannot be integrated in this type of solutions, as the IdP cannot assign the PPID based on an *unknown* RP identity.

There is also another method that prevents both IdP and RP from knowing user’s trace based on zero-knowledge algorithm, such as EL PASSO [7] and UnlimitID [8]. In such systems, the user needs to keep a secret *s* and requires IdP to generate an identity proof for blinded *s*. Then user must prove that she is the owner of *s* without exposing *s* to RP. It can protect user from both IdP-based login tracing and RP-based identity linkage. However, whenever a user tries to visit RP at a new device, she must import *s* into this device. Thus, it is not convenient for user’s login on multiple devices.

## Extended Related Works

**Security analysis of SSO systems.** Various attackers were found accessing the honest user’s account at RP by multiple methods. Some attackers exploit the vulnerabilities of user’s platforms to steal identity proof (or cookie) [14], [15]. Some other attackers temper the identity proof to impersonate an honest user, such as XSW [13], RP’s incomplete verification [14], [16], [18], and IdP spoofing [17], [18]. Some IdP services did not bind the identity proof with a corresponding RP (or the honest RP did not verify the binding), so that malicious RP can leverage the received identity proof to access to another RP [16]–[18]. The formal analysis is also used to guarantee the security of SSO protocols [11], [12].

**Authentication systems built based on Intel SGX.** Intel SGX has been used in enhancing the security and privacy of authentication systems. Rafael et al. [20] offer the credential protection approach based on SGX, which prevents an adversary from stealing a user’s credential at server side. P2A [21] is proposed to protect user’s privacy. It enables a user to generate an identity proof locally while the registration, update, freeze/thaw, and deletion of identities are managed in a blockchain. Therefore, a user can visit a service without exposing his real identity.

# THREAT MODEL

In this section, we describe the threat model and assumptions of the entities in UP-SSO.

**Adversary Goal:** (1) The adversaries attempt to impersonate a user to log into an RP (*breaking the security of SSO*); (2) the adversaries want to track a user’s login trace (*breaking the privacy*).

**Adversary Capacity:**

* **To break the security.** An adversary could act as a malicious user or a malicious RP. The adversary could control the software running outside the enclave, capturing and tempering the message transmissions, decrypting and tempering the HTTPS flows, tempering the script code running on the browser. Moreover, the attacker might also act as an external attacker, who collects all network traffics. The adversary could also allure the user to download the malicious script on his browser.
* **To break the privacy.** An adversary can act as the malicious RP and curious but honest IdP. The malicious RP may manipulate all the messages generated and trans- mitted by RP. However, the honest IdP must process the requests of RP registration and identity proof correctly, provide honest script, and never collude with others. Both RP and IdP can store and analyze the received messages.

We assume the honest user’s device is secure, for example, user would not install malicious application on his device. The application and data inside the enclave are never tempered or leaked, even the device is owned by an adversary. Moreover, the enclave application always runs the processes same as IdP server expected, as it is guaranteed by remote attestation.

# Design of UP-SSO

In this section, we will provide the detailed protocol.

## Initialization

At the very beginning, RP needs to register its domain, RP name and other essential attributes at IdP, and obtains a signed Cert. Thus, the IdP can only provide the service to the qualified RP by checking the ownership of a Cert inside the enclave application.

Moreover, user should install the enclave application on his device before the visit targeting an RP.

## Login Process

Following, we provide the detailed process of each authentication flow shown as Figure 2. It can be split into three phases, scripts downloading, PPIDs generation and identity proof issuing. To be noticed that, all the communication flows among user, RP server and IdP server are protected by TLS.

**Scripts downloading.** This phase is for the user’s browser to download the scripts from the RP and IdP. The SSO process is started with the user’s visit to an RP at his browser, and the browser downloads the RP script (step 1). Then the RP script opens a new window targeting RP login endpoint (step 2, 3). After that the user is redirected to the IdP server (step 4). Finally, the user retrieves the IdP script (step 5).

It must be noticed that, the user cannot visit IdP at step 2,3 directly. While the script in origin A opens a new window with origin B, the HTTP request to B’s server will carry the key-value *Referer: A* in the header. Thus, the RP’s domain is exposed to IdP.

**PPIDs Generation.** In this phase, IdP script invokes the enclave application to generate the PPID. At first, the IdP authenticates the user (step 6). After that, IdP script informs RP window (step 7), and RP script sends its Cert to IdP script (step 8). Then IdP script asks for user’s consent to visit targeting RP, and it invokes the enclave application (step 9). The enclave application conducts remote attestation at its initialized execution, and negotiates a symmetric key *K* with IdP server (step 10). Following the enclave application generates the *UID* request by encrypting request information with *K* (step 11). The request is sent to IdP script (step 12) and transmitted to IdP server (step 13). IdP encrypts *UID* with *K* (step 14), and transmits it to enclave application through IdP script (step 15, 16). While receiving the encrypted *UID*, the enclave application derives the *UID*, verifies the Cert and achieves RP’s domain from it, generates the symmetric key *S*, encrypts the RP’s domain as the *PPIDRP*, encrypts the hash of RP’s domain and *UID* as the *PPIDU*, and encrypts *PPIDRP* and *PPIDU* with *K* (step 17). Finally, the encrypted IDs and *S* are sent to IdP script (step 18).

The *PPIDU* must be the encrypted hash of RP’s domain and *UID*, instead of the plain digest. It avoids the IdP to find out multiple logins targeting the same RP or not, as the hash of same RP’s domain would be always the same.

**Identity proof issuing.** In this phase, IdP issues an identity proof containing the encrypted *PPIDRP* and *PPIDU*. And the RP verifies the token. At the beginning user sends the encrypted *PPIDRP* and *PPIDU* to IdP for identity proof (step 19). Then the IdP server derives the *PPIDRP* and *PPIDU*, signs the *Token* consisted of *PPIDRP* and *PPIDU* as the identity proof (step 20), and sends it to IdP script (step 21). The IdP script then sends the *Token* and s to the origin RP’s Domain through *postMessage* (step 22), and RP script uploads them to RP server (step 23). The RP server verifies the signature with IdP’s public key, compares the *PPIDRP* carried with identity proof with self-generated one, and derives the constant user account from PPIDU (step 24). At the end, RP returns the login result back to user (step 25).

# SECURITY ANALYSIS

This section presents the analysis of security and privacy.

## Security of UP-SSO

A secure SSO service ensures that only the legitimate user logins to an honest RP under his account.

**Definition 1.** An SSO system is secure *iff* the honest user's account at an honest RP would never be accessible to any adversaries.

**Theorem 1.** When the IdP is honest, the following properties of an identity token which is verified by an RP to derive the account in a login instance, result in a secure SSO system:

* **RP designation.** The designated RP is specified in identity proof, so that this identity proof is only accepted by this RP. Otherwise, the adversary may act as an RP, and collect honest users’ identity proofs to visit other RPs.
* **Integrity.** Only the IdP can generate identity proofs, and the identity proof with any modification would not be accepted.
* **Confidentiality.** Identity proof is only transmitted to the user and the designated RP.

*Proof.* An adversary can only visit the RP in the honest user's account, while it uploads an identity token to the RP and RP derives the honest user's account from it.

Although this kind of attack can also be conducted by stealing the honest user's cookie, it is not considered here due to the same origin policy. The adversary can only achieve an identity token in three ways, (a) achieving identity token from a user, (b) asking for the identity token from IdP as a user, and (c) forging or tempering the identity token itself. However, these attempts of attack are not possible as long as the requirements in the theorem are satisfied. The *confidentiality* and *RP designation* guarantees that any token received from an honest user would not be accepted by any other honest RPs. The identity token issued for the adversary can only represent the adversary's identity. The *integrity* guarantees that the adversary cannot forge or temper the identity token.

**Theorem 2.** UP-SSO is a secure SSO system, if the IdP is honest.

*Proof.* According to the Theorem 1 and Definition 1, the UP-SSO is a secure is secure, as long as it owns the required properties.

**RP designation.** Different from identity proof in OIDC system, the proof issued by UP-SSO IdP contains the encrypted RP ID instead of plain one. We consider an adversary can get an identity proof, while the domain may belong to a malicious RP (*PPIDRP* is the encrypted domain, and *PPIDU* is the encrypted hash of the domain and user’s UID, while the key is generated by enclave application). There is no chance to find key *S*’ satisfying that, the honest RP can derive its domain from *PPIDRP*, and derive valid user account from *PPIDU* with the same *S*’, as the user account must equals to the hash of honest RP’s domain and honest user’s UID.

**Integrity of identity proof.** An adversary may temper the identity proof by offering malicious key at step 22 in Figure 2. We consider only the IdP server and enclave application are honest. An adversary can get an identity proof, while the *PPIDRP* and *PPIDU* are generated with malicious user’s UID and any registered RP’s domain. There is no chance an adversary can get a key *S*’ and the domain satisfying that, the honest RP can derive its domain from *PPIDRP*, and derive the valid user account from *PPIDU* with the same *S*’.

Moreover, an adversary may try to lead IdP to issue a token with malicious PPID to break the integrity indirectly. PPID is generated based on Domain and *UID*, however, the domain is guaranteed by the Cert and *UID* is protected by the key *K*. Thus, the generation of *PPIDU* cannot be undermined. Moreover, the *PPIDU* is protected by *K* during the transmission, so that it cannot be modified. Therefore, the integrity of identity proof is guaranteed.

**Confidentiality of identity proof.** To guarantee the identity proof is securely transmitted, TLS is adopted to protect the transmissions between servers and user. Moreover, the *postMessage* restricted with an origin is adopted to protect the transmissions between RP and IdP scripts. That is, while the IdP script receives an identity proof from IdP server, it would invoke the *postMessage* function provided by browser. And the target’s origin is set as the value of RP’s domain used in generating *PPIDRP* and *PPIDU*. Browser guarantees that only the script belongs to the given origin can receive this identity proof.

So, it is proved that the UP-SSO system satisfies the sufficient requirements of a secure SSO system. Therefore, UP-SSO is proved secure.

## Privacy of UP-SSO

According to Section IV, we can find that a curious IdP can only obtain the *PPIDRP* related to the RP’s identity. The *PPIDRP* is the transformed RP domain, which is encrypted with a one-time key. Therefore, the IdP cannot know the real RP’s identity or link multiple logins on the same RP. So, the IdP-based identity tracing is not possible in UP-SSO system.

Similarly, the RP can only obtain the *PPIDU* from IdP. A malicious RP cannot derive the real user’s *UID* from the *hash(Domain, UID)*. Thus, the collusive RPs are also unable to link the same user based on *UID*. Although the malicious RP can control the RP’s Domain to lead the IdP generate incorrect *PPIDU*, it still fails to accomplish the attack. Because due to the proof of confidentiality of identity proof, once the RP’s Domain does not consist with the RP’s origin, the RP script would be unable to receive the identity proof. Therefore, the attack is not available.

# IMPLEMENTATION AND EVALUATION

We have implemented a prototype of UP-SSO. In this section, we describe the details of implementation and compare it with the open-source OIDC project to show its practical value.

## Implementation

We adopt SHA-256 for digest generation, RSA-2048 for signature generation, and 128-bit AES/GCM algorithm for symmetrical encryption.

IdP and RP servers are implemented based on Spring Boot, a popular web framework. We use the open-source auth0/java-jwt library to generate and verify the identity proof. The cryptographic computations of servers are implemented using API provided by JDK. Native messaging [19] (enables communication between chrome extension and native application) and chrome extension are adopted to enable the JavaScript code to invoke the native application installed on user’s device. Enclave application is built based on Intel enclave SDK, which offers the APIs for cryptographic computations.

## Evaluation

**Environment.** We deploy the IdP server on the device with Intel i7-8700 CPU and 32 GB RAM memory, running Windows 10 x64 system, and the RP server on the laptop with Intel i7- 4700HQ CPU and 16 GB RAM memory, running Windows 10 x64 system. The browser is chrome deployed on the IdP device. The devices are connected through the 1Gbps network.

**Result.** We have run SSO process on our prototype system 100 times, and the average time is 171 ms (from user visiting an RP to RP returning the authentication result). As the remote attestation overhead is only counted once the enclave application is initiated, the overhead is not included (about 1139 ms). For comparison, we run MITREid Connect, a popular open-source OpenID Connect project. The time cost of MITREid Connect is 113 ms. The overhead is modest.

# CONCLUSION

We propose UP-SSO, the extended PPID-based SSO system with comprehensive protections against both the IdP-based login tracing and the RP-based identity linkage. To prevent the IdP from knowing a user’s login trace in an SSO system, we split the IdP service into two parts, and shift the PPID generation from server to user-controlled environment. This part of service runs on the user’s device protected by the Intel SGX. The integrity is guaranteed through the remote attestation, even the malicious user cannot control the user-side service. We systemically analyze the UP-SSO protocol and guarantee its security. We have implemented the prototype of UP-SSO system, and the performance evaluation on the prototype demonstrates the overhead is modest.

##### References

1. “Thetop500sitesontheweb,”https://www.alexa.com/topsites,accessed July 30, 2019.
2. P. A. Grassi, M. Garcia, and J. Fenton, “NIST special publication 800-63c digital identity guidelines: Federation and assertions,” *National Institute of Standards and Technology, Los Altos, CA*, 2017.
3. N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, “OpenID Connect core 1.0 incorporating errata set 1,” *The OpenID Foundation, specification*, 2014.
4. T. Hardjono and S. Cantor, “SAML v2.0 subject identifier attributes profile version 1.0,” *OASIS standard*, 2019.
5. D. Fett, R. Ku ̈sters, and G. Schmitz, “SPRESSO: A secure, privacy- respecting single sign-on system for the web,” in *ACM CCS, Denver, CO, USA*, 2015, pp. 1358–1369.
6. D. Fett, R. Küsters, and G. Schmitz, “Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the web,” in *ESORICS*, 2015, pp. 43–65.
7. Z. Zhang, M. Kro ́l, A. Sonnino, L. Zhang, and E. Rivie`re, “EL PASSO: efficient and lightweight privacy-preserving single sign on,” *Proc. Priv. Enhancing Technol.*, vol. 2021, no. 2, pp. 70–87, 2021.
8. M. Isaakidis, H. Halpin, and G. Danezis, “Unlimitid: Privacy- preserving federated identity management using algebraic macs,” in *ACM WPES@CCS 2016, Vienna, Austria*, E. R. Weippl, S. Katzen- beisser, and S. D. C. di Vimercati, Eds. ACM, 2016, pp. 139–142.
9. V. Costan and S. Devadas, “Intel sgx explained.” *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.
10. A. Armando, R. Carbone, L. Compagna, J. Cue ́llar, and M. L. Tobarra, “Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps,” in *ACM FMSE, Alexandria, VA, USA*, 2008, pp. 1–10.
11. D. Fett, R. Ku ̈sters, and G. Schmitz, “A comprehensive formal security analysis of OAuth 2.0,” in *ACM CCS, Vienna, Austria*, 2016, pp. 1204– 1215.
12. R. K. Daniel Fett and G. Schmitz, “The web SSO standard OpenID Connect: In-depth formal security analysis and security guidelines,” in *IEEE CSF, Santa Barbara, CA, USA*, 2017, pp. 189–202.
13. J. Somorovsky, A. Mayer, J. Schwenk, M. Kampmann, and M. Jensen, “On breaking SAML: Be whoever you want to be,” in *USENIX Security Symposium, Bellevue, WA, USA*, 2012, pp. 397–412.
14. R. Wang, S. Chen, and X. Wang, “Signing me onto your accounts through Facebook and Google: A traffic-guided security study of com- mercially deployed single-sign-on web services,” in *IEEE S&P, San Francisco, California, USA*, 2012, pp. 365–379.
15. A. Armando, R. Carbone, L. Compagna, J. Cue ́llar, G. Pellegrino, and A. Sorniotti, “An authentication flaw in browser-based single sign-on protocols: Impact and remediations,” *Computers & Security*, vol. 33, pp. 41–58, 2013.
16. H. Wang, Y. Zhang, J. Li, and D. Gu, “The achilles heel of OAuth: A multi-platform study of OAuth-based authentication,” in *ACSAC, Los Angeles, CA, USA*, 2016, pp. 167–176.
17. C. Mainka, V. Mladenov, and J. Schwenk, “Do not trust me: Using malicious IdPs for analyzing and attacking single sign-on,” in *IEEE EuroS&P, Saarbru ̈cken, Germany*, 2016, pp. 321–336.
18. C.Mainka,V.Mladenov,J.Schwenk,andT.Wich,“Sok:Singlesign-on security - an evaluation of OpenID Connect,” in *IEEE EuroS&P, Paris, France*, 2017, pp. 251–266.
19. “Native messaging,” https://developer.mozilla.org/en- US/docs/Mozilla/ Add-ons/WebExtensions/Native messaging, accessed May 20, 2021.
20. R. C. R. Conde, C. A. Maziero, and N. C. Will, “Using intel SGX to protect authentication credentials in an untrusted operating system,” in *IEEE ISCC, Natal, Brazil*. IEEE, 2018, pp. 158–163.
21. T. Song, W. Wang, F. Lang, W. Ouyang, Q. Wang, and J. Lin, “P2A: privacy preserving anonymous authentication based on blockchain and SGX,” in *Inscrypt, Guangzhou, China*, ser. Lecture Notes in Computer Science, Y. Wu and M. Yung, Eds., vol. 12612. Springer, 2020, pp. 257–276.