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

***Abstract*—Single sign-on (SSO) services are widely deployed in 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 in to relying parties (RPs) by submitting an *identity proof*. However, SSO introduces the potential leakage of user privacy, indicated by NIST. That is (*a*) a curious IdP could track a user's all visits to any RP 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. 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 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 correctness 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.**

I. Introduction

Single sign-on (SSO) systems, such as OAuth, OpenID Connect (OIDC) and SAML, have been widely deployed for web authentication. SSO delegates user authentication on websites from online services (Relying Party, RP) to a third party (Identity Provider, IdP). Using SSO, a user no longer needs to maintain multiple credentials for different RPs. According to Alexa, the popular web traffic analysis site, 80 websites among the top-100 websites integrate SSO services.

However, the wide adoption of SSO also raises new privacy concerns. NIST [1] indicates that curious IdP or multiple collusive RPs could break the users' privacy in such ways. (1) **IdP-based login tracing**. The IdP knows the identities of the RP and user in each single login instance; (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 [1] , OIDC [2] and SAML [3]. That is, IdP should offer a user multiple individual IDs for different RPs. Thus, an RP cannot correlate the PPID with user's PPID at another RP. More and more IdPs have adopted PPID to protect user privacy, such as Active Directory Federation Services and Oracle Access Management. And NORDIC APIS and CURITY also suggest adopting PPID in SSO to protect user privacy. However, PPID mechanism cannot protect user from IdP-based login tracing, as the PPID generation requires the participation of RP identity.

Beside of PPID mechanism, there have been many works on protecting user privacy in SSO systems. However, to the best of our knowledge, none of the existing techniques can offer the comprehensive protection of privacy or achieve it by integrating PPID. Here we give a brief introduction of existing solutions and explain the flaws of these schemes.

- **Simply hiding RP ID from IdP.** For example, SPRESSO [4] were proposed to defend against IdP-based login tracing by hiding RP ID from IdP. It uses the encrypted RP ID instead, and IdP issues the identity proof for this one-time encrypted ID. However, PPID is not available in this type of solution, as IdP cannot offer an RP a constant PPID with one-time irrelevant RP ID.
- **Proving user identity based on zero-knowledge proof.** In this type of solution, such as EL PASSO [5], the user needs to keep a secret s and requires IdP to generate an identity proof for blinded s. Then user has to prove that she is the owner of s to RP without exposing s to RP. 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.

In this paper, we propose UP-SSO, the comprehensive protection against both IdP-based login tracing and RP-based identity linkage. The key idea of UP-SSO is shifting PPID generation from server to user client. The user client takes the responsibility for generating PPID and transforming RP ID into an encrypted one-time ID, so that the IdP-based login tracing is not impossible as IdP no longer receives plain RP ID. To be noticed is that the user-side service must be deployed at the user-controlled and IdP trusted environment (Intel SGX).

Our contribution. The main contribution of UP-SSO is that, it provides the PPID-included identity token without exposing RP ID to IdP, by shifting the PPID generation from IdP server to user controlled device, and protecting it with the Intel SGX. Therefore, UP-SSO offers the comprehensive solution to hide the users' login traces from both the curious IdP and malicious collusive RPs. Moreover, it provides a new way of using secure hardware (i.e., Intel SGX) for SSO systems.

The rest of the paper is organized as follows. We first introduce the background in Section II. Then, we describe the threat model, and our UP-SSO design in Sections III and IV, followed by a systemic analysis of security and privacy

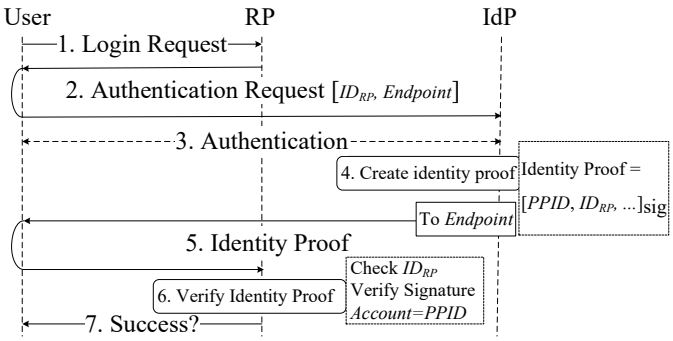

Fig. 1: The implicit protocol flow of OIDC.

in Section V. We provide the implementation specifics and experiment evaluation in Section VI, discuss the related work in Section VII, and conclude our work in Section VIII.

II. Background

UP-SSO is compatible with OIDC, and achieves privacy protection based on the SGX. Here, we provide a brief introduction on OIDC and the SGX.

A. OpenID Connect and PPID

OIDC [2] is an extension of OAuth 2.0 to support user authentication, and has become one of the most prominent SSO authentication protocols.

Implicit flow of user login. OIDC supports three processes for the SSO authentication session, known as *implicit flow*, *authorization code flow* and *hybrid flow* (i.e., a mix-up of the previous two). Here, we choose the OIDC implicit flow as the example to illustrate the protocol.

As shown in Figure 1, At the beginning, user attempts to log in to an RP (step 1). Then RP constructs a request for identity proof, which is redirected by the user to the corresponding IdP. The request contains ID_{RP}, RP's endpoint other attributes (step 2). If the user has not been authenticated yet, the IdP performs an authentication process (step 3). Following, IdP signs the ID_{RP} and PPID as the identity token (step 4), and sends it back to the RP (step 5). Finally, the RP verifies the id token, extracts user identifier (step 6), and returns the authentication result to the user (step 7).

Pairwise Pseudonymous Identifier (PPID). PPID is the user identifier provided by IdP identifying a user. Different from the normal user ID, PPID is an RP-specific user ID. That is, while a user visits different RPs, IdP would provide different but constant user IDs (i.e., PPID) for these RPs. NIST [1] suggests that PPID should be either generated randomly and assigned to users by the IdP, or derived from other user's information if the derivation is done in an irreversible, unguessable manner.

B. Intel SGX

Intel Software Guard Extensions (Intel SGX) [6] is the hardware-based security mechanism provided by Intel processors. Enclaves [6] and remote attestation [6] mechanism are offered by Intel SGX.

Enclave. The enclave's code and data is 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.

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.

III. Threat Model

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

Adversaries' Goal: (1) The adversaries can impersonate a user to log in to an RP (**breaking security**); (2) the adversaries can track a user's login trace (**breaking privacy**).

Adversaries' Capacities:

- **An adversary can act as the malicious user.** The adversary can control the software running outside the enclave, capturing and tempering the message transmission between enclave application and another entities, decrypting and tempering the HTTPS flows, tempering the script code running on the browser.
- **An adversary can act as the malicious RP.** The adversary can lead the user to log in to the malicious RP. In this situation, the adversary can manipulate or all the messages transmitted through RP, and collect all the flows received from user to link the user identity.
- **An adversary can act as the curious but honest IdP.** The curious IdP can store and analyze the received messages. However, the honest IdP must process the requests of RP registration and identity proof correctly, provide honest script, and never collude with others.
- **External attackers.** An adversary can collect all the network flows. The adversary can also lead the user to download the malicious script on her browser.

We assume the honest user's device is secure, for example, user would not install malicious application on her device.

IV. Design of UP-SSO

In this section, we will provide the detailed protocol shown as Figure 2.

RP Registration (once for each RP). 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.

Following, we provide the detailed process of each authentication flow. It can be split into three phases, scripts downloading, PPID generation and identity token issuing.

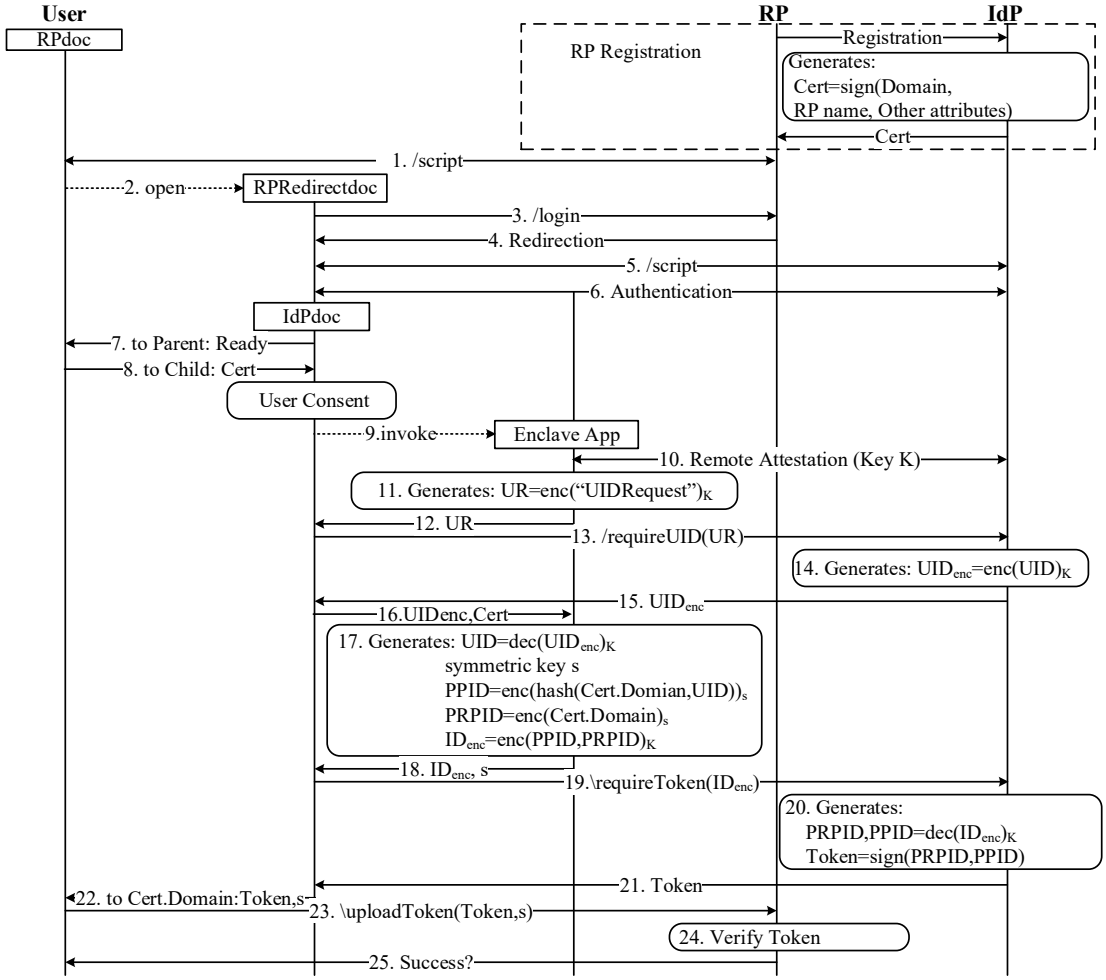

Fig. 2: The protocol flow of UP-SSO.

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 her 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 $Referer : A$. Thus, the RP's domain is exposed.

PPID 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 sends the ready signal to RP window (step 7), and RP script sends its Cert to IdP script (step 8). Then IdP script asks for user's consent to log in to targeting RP, and then 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 (i.e., UR) by encrypting request information with K (step 11). UR 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 with K, verifies the Cert, achieves RP's domain from Cert, generates the symmetric key s, encrypts the RP's domain as the transformed RP ID (i.e., $PRPID$), encrypts the hash of RP's domain and UID as the $PPID$, and encrypts $PRPID$ and $PPID$ with K (step 17). Finally enclave application sends the encrypted IDs and s to IdP script (step 18).

The PPID 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 token issuing. In this phase, IdP issues an identity token containing the encrypted $PPID$ and $PRPID$. And the RP verifies the token. At the beginning user sends the encrypted $PRPID$ and $PPID$ to IdP for identity token (step 19). Then the IdP server derives the $PRPID$ and $PPID$, signs the $token$ consisted of $PRPID$ and $PPID$ 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 $PRPID$ carried with identity token with self-generated one, and derives the constant user account from $PPID$ (step 24). At the end, RP returns the login result back to user (step 25).

V. Security Analysis

This section presents the analysis of security and privacy.

A. Security of UP-SSO

We summarize the basic requirements of SSO systems based on existing theoretical analyses [7]–[9] and practical attacks [10]–[16]. These requirements enable an SSO system to provide secure authentication services for RPs.

- **User identification.** When a user logs in to an RP multiple times, the RP extracts the identical user identifier from identity proofs.
- **RP designation.** The designated RP is specified in identity proof, so that this identity proof is only accepted by this RP.
- **Integrity and confidentiality.** Only the IdP can generate identity proofs, and the identity proof with any modification would not be accepted. Meanwhile, identity proof is only transmitted to the user and the designated RP.

Following, we would describe the security mechanisms provided in UP-SSO system.

User identification. UP-SSO does satisfy the requirement as it provides the PPID, from which the RP can derive a constant user account (i.e., the hash of PR domain and UID).

RP designation. Different from identity token in OIDC system, the token issued by UP-SSO IdP contains the encrypted RP ID instead of plain one. We consider an adversary can get an identity token including
$PRPID = enc(\texttt{MaliciousDomain})_{key}$ and
$PPID = enc(hash(\texttt{MaliciousDomain}, UID))_{key}$.
There is no chance an adversary can find key' satisfying
$dec(PRPID)_{key'} \equiv \texttt{HonestDomain}$ and
$dec(PPID)_{key'} \equiv hash(\texttt{HonestDomain}, UID)$.

Integrity of identity token. An adversary may temper the identity token 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 token including
$PRPID = enc(\texttt{AnyDomain})_{key}$ and
$PPID = enc(hash(\texttt{AnyDomain}, \texttt{MaliciousUID}))_{key}$.
The $MaliciousUID$ must belong to the adversary, while the `AnyDomain` can be any registered RP domain. There is no chance an adversary can get key' satisfying that,
$dec(PRPID)_{key'} \equiv \texttt{HonestDomain}$ and
$dec(PPID)_{key'} \equiv hash(\texttt{HonestRPDomain}, \texttt{HonestUID})$.

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 $PPID$ cannot be undermined. Moreover, the $PPID$ is protected by K during the transmission, so that it cannot be modified. Therefore, the integrity of identity token is guaranteed.

Confidentiality of identity token. Confidentiality of identity token requires that the identity token must be transmitted to RP correctly without being exposed to adversaries.

To guarantee the identity token is securely transmitted, $postMessage$ restricted with an origin is adopted in UP-SSO system. That is, while the IdP script receives an identity token from IdP server, it would invoke the $postMessage$ function provided by browser. And the target's origin is set as the value $RPDomain$ used in generating $PPID$ and $PRPID$. Browser guarantees that only the script belongs to the set origin can receive this identity token.

B. Privacy of UP-SSO

According to Section IV, we can find that a curious IdP can only obtain the $PRPID$ related to the RP's identity. The $PRPID$ 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 $PPID$ 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 $PPID$, it still fails to accomplish the attack. Because due to the proof of confidentiality of identity token, once the RP's $Domain$ is not consist with the RP's origin, the RP script would be unable to receive the identity token. Therefore, the attack is not available.

VI. 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.

A. 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 token. The cryptographic computations of servers are implemented using API provided by JDK. Native messaging [17] (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.

B. Evaluation

Environment. In the evaluation, we deploy the RP and IdP servers on the device with Intel i7-8700 CPU and 32GB RAM

memory, running Windows 10 x64 system. The browser is Chrome v90.0.4430.212, deployed on the same device.

Result. We have run SSO process on our prototype system 100 times, and the average time is 154 ms (excluding the overhead of remote attestation). For comparison, we run MITREid Connect, a popular open-source OpenID Connect implementation in Java. The time cost of MITREid Connect is 113 ms. The overhead is modest.

VII. 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) [11], [12]. Some other attackers temper the identity token to impersonate an honest user, such as XSW [10], RP's incomplete verification [11], [13], [15], and IdP spoofing [14], [15]. Some IdP services did not bind the identity token with a corresponding RP (or the honest RP did not verify the binding), so that malicious RP can leverage the received identity token to access to another RP [13]–[16]. The formal analysis is also used to guarantee the security of SSO protocols [8], [9].

Privacy-preserving SSO systems. NIST [1] 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 [4] (using encrypted RP identifier) and BrowserID [18] (RP identifier is added by user). However, they are not defensive to RP-based identity linkage, and cannot integrate PPID. 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 [5] and UnlimitID [19]. However, this type of solution requires user should remember a secret representing her identity, which is not convenient for login on multiple devices.

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 her real identity.

VIII. Conclusion

In this paper, we propose UP-SSO, the extension of PPID SSO system with comprehensive protection against both IdP-based login tracing and 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 introduced overhead is modest.

References

[1] 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.

[2] 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.

[3] T. Hardjono and S. Cantor, "SAML v2.0 subject identifier attributes profile version 1.0," *OASIS standard*, 2019.

[4] D. Fett, R. Küsters, 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.

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

[6] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[7] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, 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.

[8] D. Fett, R. Küsters, and G. Schmitz, "A comprehensive formal security analysis of OAuth 2.0," in *ACM CCS, Vienna, Austria*, 2016, pp. 1204–1215.

[9] 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.

[10] 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.

[11] R. Wang, S. Chen, and X. Wang, "Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services," in *IEEE S&P, San Francisco, California, USA*, 2012, pp. 365–379.

[12] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, 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.

[13] 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.

[14] 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, Saarbrücken, Germany*, 2016, pp. 321–336.

[15] C. Mainka, V. Mladenov, J. Schwenk, and T. Wich, "Sok: Single sign-on security - an evaluation of OpenID Connect," in *IEEE EuroS&P, Paris, France*, 2017, pp. 251–266.

[16] R. Yang, W. C. Lau, J. Chen, and K. Zhang, "Vetting single sign-on SDK implementations via symbolic reasoning," in *USENIX Security, Baltimore, MD, USA*, 2018, pp. 1459–1474.

[17] "Native messaging," https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Native_messaging, accessed May 20, 2021.

[18] 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.

[19] 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. Katzenbeisser, and S. D. C. di Vimercati, Eds. ACM, 2016, pp. 139–142.

[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.