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. After authenticated by the identity providers (IdPs), a user is allowed to log in to relying parties (RPs) by submitting an *identity proof* (i.e., id token of OpenID Connect or SAML assertion). 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 multiple individual IDs for different RPs. Popular SSO protocols, such as OpenID Connect and SAML, have already provided PPID as the optional configuration. It is suggested by many identity service providers, such as NORDIC APIS and CURITY, PPID should be adopted in all SSO systems. However, PPID mechanism cannot protect user from IdP's tracking, as the generation of PPID requires the participation of RP ID
.

In this paper, we propose an SSO system, Enhancing the User Privacy of SSO by Integrating PPID and SGX, named UP-SSO. UP-SSO provides the enhanced PPID mechanism to protect a user's activity profile of RP visits from both the curious IdP and the collusive RPs. It separates a 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. User client also transforms RP ID into one-time encrypted ID, used by IdP for identity token generation, instead of RP's real ID. The correctness of client can be verified by IdP through remote attestation. The server-side service running on the IdP server still takes the responsibility of authenticating users, storing the private key, issuing the identity token and etc. The detailed design of UP-SSO is described in this paper, and the systemically analysis is provided to guarantee its security. We implemented the prototype system of UP-SSO, and the evaluation of prototype system shows the overhead is modest.

Keywords: SSO · Privacy · Intel SGX.

1 Introduction

Single sign-on (SSO) systems, such as OAuth, OpenID Connect and SAML, have been widely deployed nowadays as a convenient web authentication mechanism.

SSO delegates user authentication on websites to a third party, so-called identity providers (IdPs), so that users can access different services on cooperating sites, so-called relying parties (RPs), via a single authentication process. Using SSO, a user no longer needs to maintain multiple credentials for different RPs, instead, she only maintains the credential for the IdP, which further provides identity proofs to RPs. For RPs, SSO shifts the burden of user authentication to IdPs and therefore reduces their own security risks and costs. As a result, SSO has been widely integrated with modern web systems. According to Alexa [8], 80 websites among the top-100 websites and 6.3% [20] of the Top-1M websites integrate SSO services. Meanwhile, many email and social networking providers such as Google, Facebook, Twitter, etc. have been actively serving as social identity providers to support social login.

However, the wide adoption of SSO also raises new privacy concerns. Currently, more and more companies are interested in the users' online trace for business practices, such as accurate delivery advertising. For example, large internet service providers, such as Google and Facebook, are interested in collecting users' online behavioral information for various purposes (e.g., Screenwise Meter [28] and Onavo [14]). Now, by simply serving the IdP role, these companies can easily collect a large amount of data to reconstruct users' online traces. NIST [21] indicates that curious IdP or multiple collusive RPs could break the users' privacy as follows.

- *IdP-based login tracing.* The IdP knows the identities of the RP and user in each single login instance, to generate the identity proof. As a result, a curious IdP could discover all the RPs that the victim user attempts to visit and profile her online activities.
- *RP-based identity linkage.* The RP learns a user's identity from the identify proof. When the IdP generates identity proofs for a user, while the same user identifier is used in identity proofs generated for different RPs, malicious RPs could collude to not only link the user's login activities at different RPs for online tracking but also associate her attributes across multiple RPs.

The IdP-based login tracing and RP-based identity linkage would bring the real threat to users. Imagine that, a user concerning her privacy would avoid to leave her full sensitive information at an application. She only leaves parts of her sensitive messages at each applications, for example, using real name on a social website and address on shopping website. And she would try not to leave any linkable message to avoid applications combining her informations, such as email. However, the privacy leaks in SSO systems make her effort in vain. As long as a user employs the SSO system, such as Google Account, to log in to these applications, the applications providers and Google can combine her information based on the SSO account.

To protect user privacy, the Pairwise Pseudonymous Identifier (PPID) is suggested by NIST [21] and popular SSO protocols, such as OpenID Connect (OIDC) [33] and SAML [23]. In such SSO system, IdP offers a user multiple individual IDs for different RPs. As a result, an RP cannot correlate the received PPID with the user's PPID at another RP. Thus, collusive RPs cannot link a user's logins targeting different RPs. Recently more and more IdPs have adopted PPID to protect user privacy. For example, Active Directory Federation Services and Oracle Access Management support the use of PPID [3, 7]. And identity service providers such as NORDIC APIS and CURITY suggest adopting PPID in SSO to protect user privacy [2, 6].

There have already 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 for privacy-preserving SSO and explain the flaws of these schemes.

- *Simply hiding RP ID from IdP.* For example, SPRESSO [17] 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 [45] and UnlimitID [25], 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, it is not convenient for user login on multiple devices. For security consideration, the s must be too long for user to remember. And the user's identity is associated with the private s, therefore, if the user wants to log in to the RP on a new device, she must import the large s into this device.
- *Completely anonymous SSO system.* Anonymous SSO scheme is proposed to hide the user's identity to both the IdP and RPs in many manners. For example, in the anonymous SSO [22], it allows a user to visit the RP without exposing her identity to both IdP and RP based on zero-knowledge proof. However, it can only be applied to the anonymous services that do not identify the user, but not available in current personalized internet service.

In this paper, we propose UP-SSO, Enhancing the User Privacy of SSO by Integrating PPID and SGX, the comprehensive protection against bosh IdP-based login tracing and RP-based identity linkage. The key idea of UP-SSO is shifting part of IdP service (i.e., PPID generation), from server to user client. The user-side IdP service takes the responsibility for generating PPID and transforming RP ID into an encrypted one-time ID (as an identity proof must contain the RP ID to avoid misuse of token [30, 31, 38, 42]). For IdP, it can only obtain a encrypted one-time RP ID, so that the IdP-based login tracing is not impossible. To be noticed is that the user-side service must be deployed at the user controlled and IdP trusted environment, i.e., Intel SGX. With SGX, the secure hardware supported by intel CPU, the user-side service can be deployed at user's PC, and it can be verified by IdP through *Remote Attestation*, the function provided by SGX. Moreover, other essential IdP services, such as user authentication and identity proof issuing, must be deployed at server side for security consideration.

There have already been efforts on implementing authentication with security hardware, such as FIDO [29, 37]. It enables a secure hardware (e.g., a smartphone) to store user's private key while the RP server stores the public key. Therefore, a user can be authenticated by the RP through a signed login challenge. However, compared with UP-SSO (as well as traditional SSO), FIDO does not provide an authority center, such as IdP. It brings about the problem that, the FIDO service cannot set accessibility limits without additional user information. For example, an RP requires that all the visitors must be adults. In UP-SSO, the user only needs to be verified at IdP and the result can be included in an identity token. However, in FIDO system, the verification is not available except that user uploads specific information, which would link the user's accounts in different RPs.

To guarantee the security of UP-SSO, we provide the description of threat model and detailed systemic analysis, that shows UP-SSO prevents all the known attackers. Moreover, we have implemented a prototype of UP-SSO, including the servers, user client and user-side JavaScript codes. And we compare the performance of the UP-SSO prototype with a state-of-the-art SSO system (e.g., OpenID Connect). The time cost of UP-SSO and OpenID Connect is, respectively, about 154 and 113 ms. The overhead is modest.

Our contribution. The main contribution of UP-SSO is that, we shift the PPID generation from IdP server to user controlled device, and protect it with the Intel SGX. It enables the IdP to provide the PPID-included identity token without exposing RP ID to IdP, while the user-side service is verified by IdP through remote attestation to guarantee its integrity. 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 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 2. Then, we describe the threat model, and our UP-SSO design in Sections 3 and 4, followed by a systemic analysis of security and privacy in Section 5. We provide the implementation specifics and experiment evaluation in Section 6, discuss the related works in Section 7, and conclude our work in Section 8.

2 Background

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

2.1 OpenID Connect and PPID

OIDC [33] is an extension of OAuth 2.0 to support user authentication, and has become one of the most prominent SSO authentication protocols. Same as other SSO protocols [23], OIDC involves three entities, i.e., *users*, the *identity provider (IdP)*, and *relying parties (RPs)*. PPID is suggested in OpenID Connect protocol to protect the user from a possible correlation among RPs.

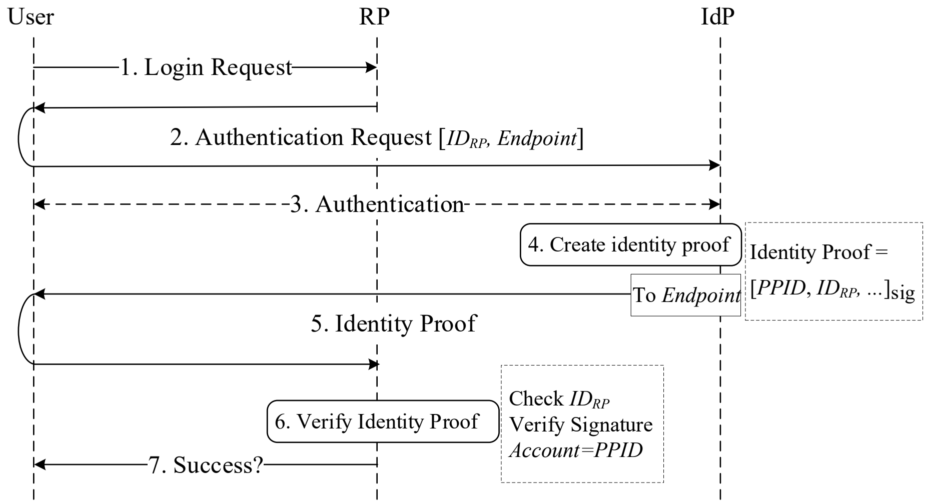

Fig. 1: The implicit protocol flow of OIDC.

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. In the implicit flow, an *id token* is generated as the identity proof, which contains a user identifier, an RP identifier, IdP's identifier, the validity period, and other requested attributes. The IdP signs the id token using its private key to ensure integrity.

As shown in Figure 1, the implicit flow of OIDC consists of 7 steps.

1 At the beginning, user attempts to log in to an RP.
2 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 and a set of requested user attributes.
3 If the user has not been authenticated yet, the IdP performs an authentication process.
4 If the RP's endpoint in the request matches the one registered at the IdP, it generates an identity token. Otherwise, IdP generates a warning to notify the user about potential identity proof leakage.
5 IdP sends the identity token back to the RP.
6 The RP verifies the id token, and extracts user identifier from the identity token.
7 Finally, RP returns the authentication result to the user.

Pairwise Pseudonymous Identifier (PPID). PPID is the user identifier provided by IdP for RP identifying a user. Different from the normal user ID, PPID is an RP-specific use ID. That is, while a user visits different RPs, IdP would provide different but constant user IDs (i.e., PPID) for these RPs. NIST [21] suggests that PPID should be used to prevent the user's account at the IdP from being easily linked at multiple RPs through use of a common identifier. It also 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 (e.g., using a keyed hash function with a secret key). For example, we have learned that, MITREid Connect [4], a popular open source OIDC project, generates PPID using a secure random number generator, and stores the key-value map of corresponding RP and PPID.

2.2 Intel SGX

Intel Software Guard Extensions (Intel SGX) [13] is the hardware-based security mechanism provided by Intel since the sixth generation Intel Core microprocessors. It offers memory encryption that isolates specific application code and data in memory, that allows user-level code to allocate private regions of memory, called enclaves [13], which guarantees the running code are well protected from the adversary outside the enclave.

Enclave. The enclave's code and data is stored in Processor Reserved Memory (PRM). PRM is the subset of Dynamic random-access memory (DRAM), and it 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 [13] enables the software inside an enclave to attest to a remote entity that it is trusted. That is, during the attestation, the remote entity would receive an SGX attestation signature, containing the enclave's measurement (a measurement of the code and data loaded in enclave). The SGX remote attestation allows a player to verify three things, the application's identity, its intactness (that it has not been tampered with), and that it is running securely within an enclave on an Intel SGX enabled platform. Moreover, with the remote attestation, the secure key exchange between the player and remote enclave application is also available even the application runs in the malicious environment.

3 Threat Model and Assumptions

UP-SSO is compatible with OIDC, consisted of a number of RPs, user agents(including browser and enclave application) and an IdP. In this section, we describe the threat model and assumptions of these entities in UP-SSO.

3.1 Threat Model

Adversaries' Goal:

- **Breaking the security.** The adversaries can impersonate an honest user to log in to the honest RP.
- **Breaking the privacy.** The adversaries can track a user's login trace on each RP.

Adversaries' Capacities:

- **Malicious user.** An adversary can act as the malicious user. The adversary can control all the software running outside the enclave, such as capturing and tempering the message transmission among enclave application and other entities, decrypting and tempering the https flow outside the enclave, tempering the script code running on the browser. For example, a malicious user may try to temper the PPID sent to IdP to achieve an identity token representing an honest user.
- **Malicious RP.** 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 message transmitted through RP, and collect all the flows received from user to link the user identity.
- **Curious but honest IdP.** An adversary can act as the curious but honest IdP. The curious IdP can store and analyze the received messages, and perform the timing attacks, attempting to achieve the IdP-based linkage. However, the honest IdP must process the requests of RP registration and identity proof correctly, provide honest script, and never collude with others (e.g., malicious RPs and users).
- **External attackers.** Moreover, an adversary can collect all the network flow transmitted through a user. The adversary can also lead the user download the malicious script on her browser.

3.2 Assumptions

We assume the honest user's device is secure, for example, the user would not install any malicious application on her device. The application and data inside the enclave are never tempered or leaked, even in the malicious user's device.

The TLS is also adopted and correctly implemented at the system, so that the communications among entities ensure the confidentiality and integrity. The cryptographic algorithms and building blocks used in UP-SSO are assumed to be secure and correctly implemented.

Phishing attack is not considered in this paper.

4 Design of UP-SSO

The UP-SSO is compatible with OIDC, besides that the IdP service is separated into user part and server part. The user agent (i.e., user-side IdP service) would

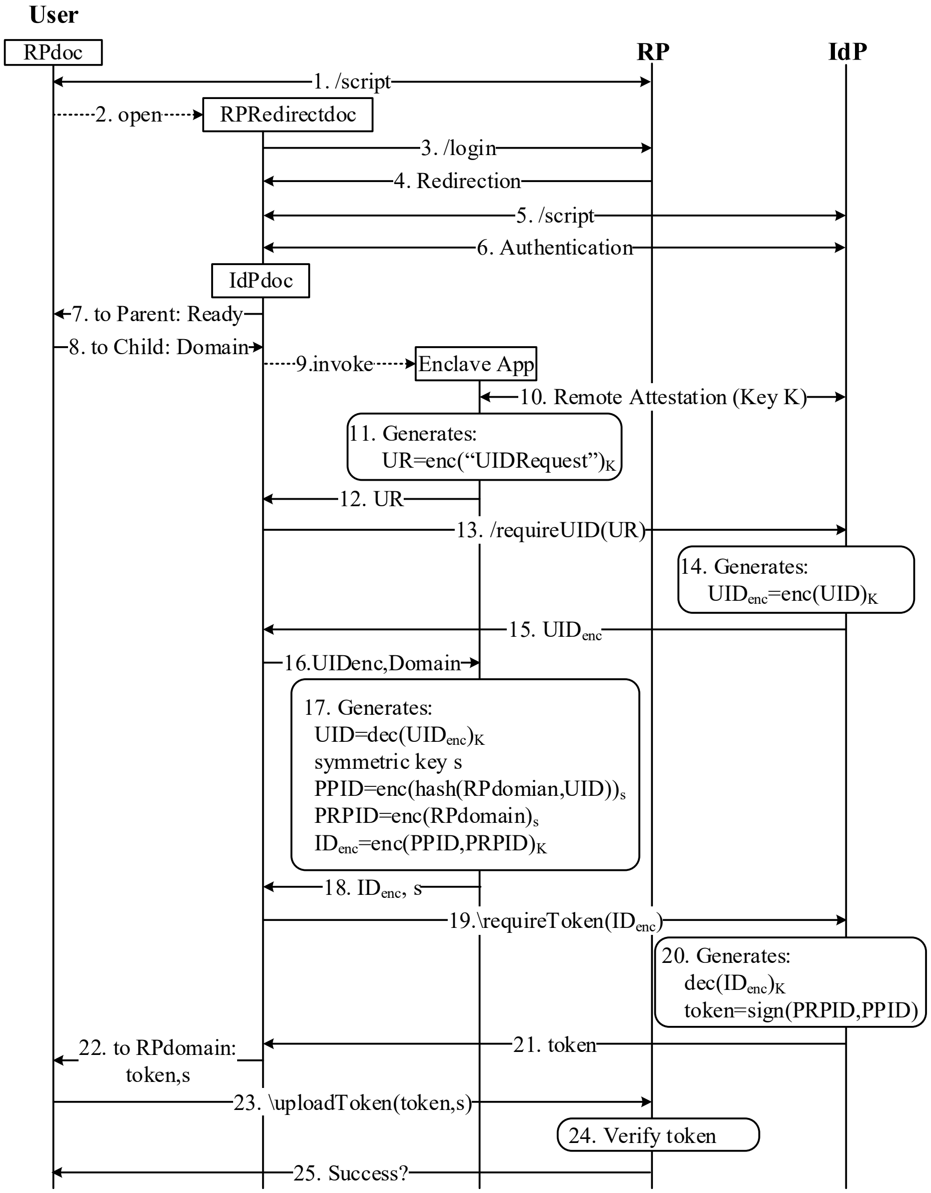

Fig. 2: The protocol flow of UP-SSO.

obtain the UID from server part service, transform UID into the PPID, and encrypt the RPID and PPID with an one-time symmetric key to avoid IdP server digging out the RP's identity. As the enclave application is protected by SGX, it must not conduct any malicious behaviour. The server-side IdP service takes the responsibility of authenticating the user, retrieving the UID for each user, and issuing the identity token (i.e., the identity proof) consisted the privacy-preserving RP and user identifier.

4.1 UP-SSO procedures

Following, we provide the detailed protocol of UP-SSO. The procedures of UP-SSO are depicted in detail in Figure 2. It can be split into three phases, scripts downloading, PPID generation and identity token issuing. Here, we describe the details.

Scripts downloading. The scripts downloading phase is for the user's browser to download the scripts from the RP and IdP servers. The scripts work as part of the user agent role.

1 The SSO process is started with the user's visit to an RP at her browser, and the browser downloads the RP script. The script is used to conduct the behaviour defined by RP at user side.
2 The RP script opens the new window.
3 The newly opened window visits RP login endpoint.
4 The user is redirected to the IdP server from RP login endpoint.
5 The user retrieves the IdP script, which is used to deal with the interaction with IdP server, RP script and enclave application.

It must be noticed that, the user cannot visit IdP at step 2,3 directly because of the $Referer$ attribute in HTTP header. 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 $Referer : A$. Therefore, the RP's domain is exposed to IdP. With HTML5, a special attribute for links in HTML was introduced, that the $ref =$ "$noreferrer$" can be used to make $Referer$ header be suppressed. However, when such a link is used to open a new window, the new window does not have a handle on the opening window (opener) any more. The handle is necessary for UP-SSO to transmit messages between RP and IdP scripts. So the redirection is adopted to avoid the HTTP header $Referer$ and make handle of the opening window available.

PPID Generation. In this phase, IdP script invokes the the enclave application to generate the PPID.

6 At first, the IdP authenticates the user.
7 After the IdP script is downloaded, it sends the ready signal to its opener (i.e., the RP window).
8 RP script sends the RP's domain back to IdP script.
9 Then the IdP script invokes the enclave application.
10 The enclave application conducts remote attestation at its initialized execution. After the remote attestation, enclave application and IdP server shares a symmetric key K.

11 The enclave application generates the UID request (i.e., UR) by encrypting request information with K.
12 UR is sent to IdP script.
13 Then the user starts the UID request to IdP.
14 IdP encrypts UID with K.
15 IdP script receives encrypted UID from IdP.
16 IdP script transmits encrypted UID and RP's domain to enclave application.
17 The enclave application derives the UID with K, generates the symmetric key s, encrypts the RP's domain with key s 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.
18 Enclave application sends the encrypted IDs and s to IdP script.

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 same.

Moreover, there are some types of parameters required in OIDC protocol to be carried in the SSO request, such as *response_type* and *scope*. In this paper, we would not mention these attributes, and only focus on the necessary parameters.

Identity token issuing. In this phase, IdP issues an identity token containing the encrypted $PPID$ and $PRPID$. And the RP verifies the token.

19 User sends the encrypted $PRPID$ and $PPID$ to IdP for identity token.
20 The IdP server derives the $PRPID$ and $PPID$, signs the *token* consisted of $PRPID$ and $PPID$ as the identity proof.
21 IdP sends identity token to enclave application.
22 The IdP script then sends the *token* and s to the origin *RPdomain* through $postMessage$.
23 The RP script uploads the *token* and key s to RP server.
24 The RP server firstly verifies the signature with IdP's public key, then generates the $PRPID$ with its domain and key s, and compared it with the one carried by *token*. If the two $PRPID$s are equalled, RP decrypts the user's account from $PPID$, and find out the related user information in its database.
25 RP returns the login result back to user.

The origin in step 22 is essential for secure identity token transmitting. It guarantees that only the script running in the RP window can receive the *token*, that avoids the man-in-the-middle attack.

5 Security Analysis

This section presents the analysis of security and privacy of UP-SSO.

5.1 Security of UP-SSO

We summarize the basic requirements of SSO systems based on existing theoretical analyses [11, 18, 19] and practical attacks [10, 30–32, 34, 35, 38, 39, 41–44, 46]. These requirements enable an SSO system to provide secure authentication services for RPs, through identity proofs.

- **User identification.** When a user logs into a certain RP for multiple times by submitting identity proofs, the RP extracts the identical user identifier from these identity proofs, to provide personalized services for this user.
- **RP designation.** The designated receiver (or RP) is specified in an identity proof, so that this identity proof is accepted by the visited RP only.
- **Integrity and confidentiality.** Only the IdP is trusted to generate identity proofs, RPs do not accept an identity proof with any modification or a forged one. Meanwhile, a valid identity proof is transmitted only to the user and the designated RP.

Although UP-SSO follows the main rules of OIDC, that RP redirects user to IdP, and IdP authenticates the user and returns an identity token back to RP. It also breaks OIDC's mechanisms satisfying the requirements of secure SSO. For example, in OIDC system, to guarantee the confidentiality of identity token, an identity token is transmitted to RP based on the HTTP redirection (i.e., 302 redirection) mechanism, and IdP guarantees the receiver's address belongs to honest RP (verifying *redirect_uri*). However, in UP-SSO, IdP do not learn the RP's identity, so that it needs to be proved that UP-SSO provides the secure mechanism to guarantee the confidentiality of identity token.

In this section, we would describe the secure 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), similarly as it in OIDC system.

RP designation. The idea of RP designation is that, IdP issues an identity token for user *Alice* to visit RP A. Once A receives this token, it cannot use the token to visit another RP B as user *Alice*.

Different from identity token in OIDC system, the token issued by UP-SSO IdP contains the encrypted RP ID instead of a plain RP ID. We consider an adversary can get a identity token including $PRPID = \mathit{encrypt}(\mathit{Domain}, \mathit{key})$ and $PID = \mathit{encrypt}(\mathit{hash}(\mathit{Domain}, UID), \mathit{key})$. The Domain belongs to an adversary controlled server. The UID belongs to an honest user. There is no chance that an adversary can find the key key' satisfying that, $\mathit{decrypt}(\mathit{encrypt}(\mathit{Domain}, \mathit{key}), \mathit{key}') \equiv \mathit{honest\ RPDomain}$ and $\mathit{decrypt}(\mathit{encrypt}(\mathit{hash}(\mathit{Domain}, UID), \mathit{key}), \mathit{key}') \equiv \mathit{hash}(\mathit{honest\ RPDomain}, UID)$.

Integrity of identity token. Although UP-SSO system offers the same identity token as it in OIDC system, an adversary can even try to temper the identity token by provide a malicious key s at step 22 in Figure 2. We consider in the SSO process, only the IdP server and enclave application are honest. An adversary can get a identity token including $PRPID = \mathit{encrypt}(\mathit{Domain}, \mathit{key})$ and $PPID = \mathit{encrypt}(\mathit{hash}(\mathit{Domain}, UID), \mathit{key})$. The Domain is controlled by adversary and can be assigned as any value. The UID must belong to the adversary. There is no chance that an adversary can get a key, making the attack available. Because the key key' must satisfy that, $\mathit{decrypt}(PRPID, \mathit{key}') \equiv \mathit{honest\ RPDomain}$ and $\mathit{decrypt}(PPID, \mathit{key}') \equiv \mathit{hash}(\mathit{honest\ RPDomain}, \mathit{honest}\ UID)$, which is not possible.

Moreover, unlike IdP in OIDC system retrieves the PPID from its database, IdP in UP-SSO system need to receive the PPID from user client. Therefore, an adversary may try to lead IdP to accept a malicious PPID to break the integrity indirectly. Here we review the Figure 2.

- An adversary may attempt to temper the $PPID$ while it is transmitted between enclave application and IdP server. However, $PPID$ is encrypted (step 17) and the key is only known to enclave application and IdP server (key exchange at step 10).
- The adversary may also try to undermine the generation of $PPID$. The $PPID$ is generated based on $\mathit{RPDomain}$ and UID (step 17). According to the description in RP designation, the $\mathit{RPDomain}$ must belong to the target RP. The UID is retrieved according to authentication result at step 6, and encrypted.
- The adversary may also try to steal an honest user's encrypted $PPID$ and UID, and set them in step 16 or step 19. However, in an honest user's device, the IdP script and enclave application are honest, so that they do not send encrypted $PPID$ and UID to any malicious party. Moreover, we consider honest user's device is secure, so that the adversary cannot steal $PPID$ and UID in the manners, such as intercepting communications in user's device.

Confidentiality of identity token. Confidentiality of identity token requires that the identity token must be transmitted to RP correctly without being exposed to adversaries. As IdP in UP-SSO system does not learn the target RP's identity, it cannot sends the identity token to RP through an HTTP 302 redirection, which is widely used in SSO systems, such as OIDC.

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 $\mathit{RPDomain}$ used in generating $PPID$ and $PRPID$. Browser guarantees that only the script belong to the set origin can receive this identity token.

Even an adversary can work as a malicious script in user's browser and conduct malicious behaviour, such as man-in-the-middle attack, it cannot steal user's identity token. That is, while an adversary succeed in tempering the $\mathit{RPDomain}$ (at step 8 in Figure 2), according to proof of RP designation, the identity token would not be accepted by an honest RP. Otherwise, while an adversary does not temper $RPDomain$, it cannot receive the identity token due to $postMessage$ mechanism.

5.2 Privacy of UP-SSO

Based on the process shown in Section 4, we can find that a curious IdP can only obtain the $PRPID$ related with the RP's identity. The $PRPID$ is the transformed RP domain, which is encrypted with an 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 obtains the $PPID$ from IdP. An malicious RP cannot derive the real user's UID from the $hash(RPDomain, UID)$. The collusive RPs are also unable to link the same user because of the same reason. Although the malicious RP can control the $RPDomain$ 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 $RPDomain$ 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.

However, the attack based on user's cookie is not considered in this paper. That is, the cookie of user may be exploited by malicious RPs to link a user at different RPs. For example, a user may visit multiple RPs at the same time. The malicious RPs may redirect the user to each other through the hidden iframe, carrying the $PPID$. This attack is also available in other user authentication systems beside of SSO system. Moreover, it can be easily detected through multiple methods, such as checking the iframe in the script, observing redirection flow through browser network tool, and detecting the redirection based on the browser extension.

6 Implementation and Evaluation

We have implemented a prototype of UP-SSO, containing an IdP server, an RP server and the user client (including the enclave application and chrome extension). In this section, we describe the details of implementation and compare it with the open-source OIDC project to show its practical value.

6.1 Implementation

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

IdP server and RP server. IdP and RP servers are implemented based on Spring Boot, a popular web framework for Java platform. We also use the open-source auth0/java-jwt library [1] to generate and verify the identity token (in the form of Json Web Token). The encryption, decryption and other cryptographic computations are implemented using API provided by Java SDK. Moreover, the servers offer user interfaces to download JavaScript codes.

Native messaging. Native messaging [5] enables the extension of an browser to invoke the native application installed on user's device. Many popular browsers have already supported native messaging, such as chrome and firefox. For Windows users, native messaging requires a user to update her Windows Registry by running IdP provided .reg profile. The profile defines the location of native application and which extension can invoke the application (e.g., identified by chrome-extension ID).

Chrome extension. To enable the JavaScript code to invoke the enclave application through native messaging, we implement the chrome extension as part of user client. Once the user downloads the script from IdP server, the extension is invoked and inject the content script into windows. The content script provides IdP script the interface to interact with enclave application.

Enclave application. Intel provides the enclave SDK for developers to develop enclave applications for C platform. Enclave SDK offers the APIs for cryptographic computations, under hardware-level protection. An enclave application contains two parts, the inside and outside SGX parts. In our implementation, the outside part only takes responsibility of transmitting data between chrome extension and inside-SGX-part enclave application.

6.2 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.

7 Related Works

7.1 Security and Privacy of SSO system

Recently, SSO systems have been the essential infrastructure of internet service. Various SSO protocols, such as OAuth, OIDC and SAML are widely adopted by Google, Facebook and other companies. There are also troubles about the security and privacy are introduced with the SSO systems.

Various attackers were found able to access the honest user's account at RP by multiple methods. Some attackers use the stolen user's identity proof (or cookie) to impersonate a user, while the attacks are based on the vulnerabilities of user's platforms [10,41]. Some other attackers try to temper the identity token to impersonate an honest user, such as XML Signature wrapping (XSW) [35], RP's incomplete verification [31, 38, 41], IdP spoofing [30, 31] and etc. In some IdP services did not bind the issued identity token with a corresponding RP (or the honest RP did not verify the binding), so that a malicious RP can leverage the received identity token to log in to another RP [30, 31, 38, 42, 42]. Moreover, automatical tools, such as SSOScan [46], OAuthTester [44] and S3KVetter [42], are also designed to detect vulnerabilities in SSO systems.

Beside of the analysis on the implemented SSO systems, the formal analysis is also used to guarantee the security of SSO protocols. Fett et al. [18, 19] analyse the OAuth 2.0 and OIDC protocols based on the expressive Dolev-Yao style model [15]. The vulnerabilities found in these analysis enable the adversary steal the identity token form SSO systems. The analysis also shows that OAuth 2.0 and OIDC are secure once these two vulnerabilities prevented.

NIST [21] suggests that the SSO should protect user from both RP-based identity linkage and IdP-based login tracing. The pairwise user identifier (e.g., PPID) is used in some SSO protocols, such as SAML [24] and OIDC [33]. Some protocols simply hide the RP's identity from IdP, such as SPRESSO [17] (using encrypted RP identifier) and BrowserID [16] (RP identifier is added by user). However, the pairwise user identifier cannot prevent IdP-based login tracing, and simply hiding RP identifier is not defensive to RP-based identity linkage. Moreover, an analysis on Persona found IdP-based login tracing could still succeed [15, 16]. 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 [45] and UnlimitID [25]. However, this type of solution requires that the user must remember a secret representing her identity, which is not convenient for login on multiple devices.

Anonymous SSO schemes enable the users to access a service (i.e. RP) with the permission from a verifier (i.e., IdP) without revealing their identities. The anonymous SSO systems are designed based on multiple methods. For example, various cryptographic primitives, such as group signature, zero-knowledge proof, etc., were used to design anonymous SSO schemes [22, 40]. The anonymous SSO schemes allow a user to obtain a unidentified token anonymously, and access RP's service with this token. However, the RP cannot classify a user's multiple logins, as the user is completely anonymous in this system. Therefore, they are not available in current personalized internet service.

7.2 Intel SGX and Remote Attestation

Intel SGX and remote attestation have been already used in many fields, particularly in authentication and building trusted environment.

Intel SGX have been used in enhancing the security and privacy of authentication systems. Rafael et al. [12] offers the credential protection approach based on SGX, which prevent an adversary from stealing a user's credential at server side. P2A [36] 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 an service without exposing her real identity.

Remote attestation is wisely adopted on building the trusted environment. Jaehwan et al. [9] proposes a method of building a safe smart home environment for IoT by pre-blocking devices based on remote attestation. Uzair et al. [26] establishes the trust between IoT devices based on the blockchain and remote attestation, while the blockchain offers a secure framework for device registration. Moreover, there is work on how to conduct the remote attestation without an assigned key or secure memory (required in Intel SGX). Ilia et al. [27] proposed the design of an attested execution processor based on RISC-V Rocket chip architecture. It does not require secure non-volatile memory, nor a private key explicitly assigned by the manufacturer.

8 Conclusion

In this paper, we propose UP-SSO, Enhancing the User Privacy of SSO by Integrating PPID and SGX, the extension of PPID SSO system with comprehensive protection against bosh 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 a 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. The server-side service takes the responsibility of authenticating users, storing the private key, issuing the identity token and etc. We systemically analyse 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. auth0/java-jwt. `https://github.com/auth0/java-jwt`, accessed May 20, 2021
2. Build gdpr compliant apis with openid connect. `https://nordicapis.com/build-gdpr-compliant-apis-with-openid-connect/`, accessed August 20, 2020
3. Introducing identity federation in oracle access management. `https://docs.oracle.com/cd/E40329_01/admin.1112/e27239/oif_1.htm`, accessed August 20, 2020
4. MITREid connect /openid-connect-java-spring-server. `https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server`, accessed August 20, 2019
5. Native messaging. `https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Native_messaging`, accessed May 20, 2021
6. Pairwise pseudonymous identifiers. `https://curity.io/resources/architect/openid-connect/ppid-intro/`, accessed August 20, 2020

7. The role of claims. https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/technical-reference/the-role-of-claims, accessed August 20, 2020
8. The top 500 sites on the web. https://www.alexa.com/topsites, accessed July 30, 2019
9. Ahn, J., Lee, I., Kim, M.: Design and implementation of hardware-based remote attestation for a secure internet of things. Wirel. Pers. Commun. **114**(1), 295–327 (2020). https://doi.org/10.1007/s11277-020-07364-5, https://doi.org/10.1007/s11277-020-07364-5
10. Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Pellegrino, G., Sorniotti, A.: An authentication flaw in browser-based single sign-on protocols: Impact and remediations. Computers & Security **33**, 41–58 (2013). https://doi.org/10.1016/j.cose.2012.08.007, https://doi.org/10.1016/j.cose.2012.08.007
11. Armando, A., Carbone, R., Compagna, L., Cuéllar, J., Tobarra, M.L.: Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps. In: Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE), Alexandria, VA, USA. pp. 1–10 (2008). https://doi.org/10.1145/1456396.1456397, https://doi.org/10.1145/1456396.1456397
12. Conde, R.C.R., Maziero, C.A., Will, N.C.: Using intel SGX to protect authentication credentials in an untrusted operating system. In: 2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018. pp. 158–163. IEEE (2018). https://doi.org/10.1109/ISCC.2018.8538470, https://doi.org/10.1109/ISCC.2018.8538470
13. Costan, V., Devadas, S.: Intel sgx explained. IACR Cryptol. ePrint Arch. **2016**(86), 1–118 (2016)
14. Cyphers, B., Kelley, J.: What we should learn from “facebook research”. https://www.eff.org/deeplinks/2019/01/what-we-should-learn-facebook-research, accessed July 20, 2019
15. Fett, D., Küsters, R., Schmitz, G.: An expressive model for the web infrastructure: Definition and application to the BrowserID SSO system. In: Proceedings of the 35th IEEE Symposium on Security and Privacy (S&P), Berkeley, CA, USA. pp. 673–688 (2014). https://doi.org/10.1109/SP.2014.49, https://doi.org/10.1109/SP.2014.49
16. Fett, D., Küsters, R., Schmitz, G.: Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the web. In: Proceedings of the 20th European Symposium on Research in Computer Security (ESORICS). pp. 43–65 (2015)
17. Fett, D., Küsters, R., Schmitz, G.: SPRESSO: A secure, privacy-respecting single sign-on system for the web. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), Denver, CO, USA. pp. 1358–1369 (2015). https://doi.org/10.1145/2810103.2813726, https://doi.org/10.1145/2810103.2813726
18. Fett, D., Küsters, R., Schmitz, G.: A comprehensive formal security analysis of OAuth 2.0. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria. pp. 1204–1215 (2016). https://doi.org/10.1145/2976749.2978385, https://doi.org/10.1145/2976749.2978385

19. Fett, D., Küsters, R., Schmitz, G.: The web SSO standard OpenID Connect: In-depth formal security analysis and security guidelines. In: Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF), Santa Barbara, CA, USA. pp. 189–202 (2017). https://doi.org/10.1109/CSF.2017.20, https://doi.org/10.1109/CSF.2017.20
20. Ghasemisharif, M., Ramesh, A., Checkoway, S., Kanich, C., Polakis, J.: O single sign-off, where art thou? An empirical analysis of single sign-on account hijacking and session management on the web. In: Proceedings of the 27th USENIX Security Symposium, USENIX Security, Baltimore, MD, USA. pp. 1475–1492 (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/ghasemisharif
21. Grassi, P.A., Garcia, M., Fenton, J.: NIST special publication 800-63c digital identity guidelines: Federation and assertions. National Institute of Standards and Technology, Los Altos, CA (2017)
22. Han, J., Chen, L., Schneider, S., Treharne, H., Wesemeyer, S.: Anonymous single-sign-on for n designated services with traceability. In: Proceedings of the 23rd European Symposium on Research in Computer Security (ESORICS), Barcelona, Spain. pp. 470–490 (2018). https://doi.org/10.1007/978-3-319-99073-6_23, https://doi.org/10.1007/978-3-319-99073-6_23
23. Hardjono, T., Cantor, S.: SAML v2.0 subject identifier attributes profile version 1.0. OASIS standard (2019)
24. Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P., Philpott, R., Maler, E.: Profiles for the OASIS Security Assertion Markup Language (SAML) v2.0. OASIS standard (2005), accessed August 20, 2019
25. Isaakidis, M., Halpin, H., Danezis, G.: Unlimitid: Privacy-preserving federated identity management using algebraic macs. In: Weippl, E.R., Katzenbeisser, S., di Vimercati, S.D.C. (eds.) Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, WPES@CCS 2016, Vienna, Austria, October 24 - 28, 2016. pp. 139–142. ACM (2016)
26. Javaid, U., Aman, M.N., Sikdar, B.: Defining trust in iot environments via distributed remote attestation using blockchain. In: Melodia, T., Ekici, E., Abouzeid, A., Chen, M. (eds.) Mobihoc '20: The Twenty-first ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, Virtual Event, USA, October 11-14, 2020. pp. 321–326. ACM (2020). https://doi.org/10.1145/3397166.3412801, https://doi.org/10.1145/3397166.3412801
27. Lebedev, I.A., Hogan, K., Devadas, S.: Secure boot and remote attestation in the sanctum processor. IACR Cryptol. ePrint Arch. **2018**, 427 (2018), https://eprint.iacr.org/2018/427
28. Li, S., Kelley, J.: Google screenwise: An unwise trade of all your privacy for cash. https://www.eff.org/deeplinks/2019/02/google-screenwise-unwise-trade/-all-your-privacy-cash, accessed July 20, 2019
29. Machani, S., Philpott, R., Srinivas, S., Kemp, J., Hodges, J.: FIDO UAF architectural overview. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html (2017)
30. Mainka, C., Mladenov, V., Schwenk, J.: Do not trust me: Using malicious IdPs for analyzing and attacking single sign-on. In: Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P), Saarbrücken, Germany. pp. 321–336 (2016). https://doi.org/10.1109/EuroSP.2016.33, https://doi.org/10.1109/EuroSP.2016.33

31. Mainka, C., Mladenov, V., Schwenk, J., Wich, T.: Sok: Single sign-on security - an evaluation of OpenID Connect. In: Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P), Paris, France. pp. 251–266 (2017). https://doi.org/10.1109/EuroSP.2017.32, `https://doi.org/10.1109/EuroSP.2017.32`
32. Mohsen, F., Shehab, M.: Hardening the OAuth-WebView implementations in Android applications by re-factoring the Chromium library. In: Proceedings of the 2nd IEEE International Conference on Collaboration and Internet Computing (CIC), Pittsburgh, PA, USA. pp. 196–205 (2016). https://doi.org/10.1109/CIC.2016.036, `https://doi.org/10.1109/CIC.2016.036`
33. Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C.: OpenID Connect core 1.0 incorporating errata set 1. The OpenID Foundation, specification (2014)
34. Shi, S., Wang, X., Lau, W.C.: MoSSOT: An automated blackbox tester for single sign-on vulnerabilities in mobile applications. In: Proceedings of the 14th ACM Asia Conference on Computer and Communications Security (AsiaCCS), Auckland, New Zealand. pp. 269–282 (2019). https://doi.org/10.1145/3321705.3329801, `https://doi.org/10.1145/3321705.3329801`
35. Somorovsky, J., Mayer, A., Schwenk, J., Kampmann, M., Jensen, M.: On breaking SAML: Be whoever you want to be. In: Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA. pp. 397–412 (2012), `https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/somorovsky`
36. Song, T., Wang, W., Lang, F., Ouyang, W., Wang, Q., Lin, J.: P2A: privacy preserving anonymous authentication based on blockchain and SGX. In: Wu, Y., Yung, M. (eds.) Information Security and Cryptology - 16th International Conference, Inscrypt 2020, Guangzhou, China, December 11-14, 2020, Revised Selected Papers. Lecture Notes in Computer Science, vol. 12612, pp. 257–276. Springer (2020). https://doi.org/10.1007/978-3-030-71852-7_17, `https://doi.org/10.1007/978-3-030-71852-7\_17`
37. Srinivas, S., Balfanz, D., Tiffany, E., Czeskis, A.: Universal 2nd factor (U2F) overview. `https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/fido-u2f-overview-v1.2-ps-20170411.pdf` (2017)
38. Wang, H., Zhang, Y., Li, J., Gu, D.: The achilles heel of OAuth: A multi-platform study of OAuth-based authentication. In: Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC), Los Angeles, CA, USA. pp. 167–176 (2016), `http://dl.acm.org/citation.cfm?id=2991105`
39. Wang, H., Zhang, Y., Li, J., Liu, H., Yang, W., Li, B., Gu, D.: Vulnerability assessment of OAuth implementations in Android applications. In: Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC), Los Angeles, CA, USA. pp. 61–70 (2015). https://doi.org/10.1145/2818000.2818024, `https://doi.org/10.1145/2818000.2818024`
40. Wang, J., Wang, G., Susilo, W.: Anonymous single sign-on schemes transformed from group signatures. In: Proceedings of the 5th International Conference on Intelligent Networking and Collaborative Systems, Xi'an, China. pp. 560–567 (2013). https://doi.org/10.1109/INCoS.2013.104, `https://doi.org/10.1109/INCoS.2013.104`
41. Wang, R., Chen, S., Wang, X.: Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services. In: Proceedings of the 33rd IEEE Symposium on Secu-

rity and Privacy (S&P), San Francisco, California, USA. pp. 365–379 (2012). https://doi.org/10.1109/SP.2012.30, https://doi.org/10.1109/SP.2012.30

42. Yang, R., Lau, W.C., Chen, J., Zhang, K.: Vetting single sign-on SDK implementations via symbolic reasoning. In: Proceedings of the 27th USENIX Security Symposium, USENIX Security, Baltimore, MD, USA. pp. 1459–1474 (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/yang
43. Yang, R., Lau, W.C., Shi, S.: Breaking and fixing mobile app authentication with OAuth2.0-based protocols. In: Proceedings of the 15th International Conference on Applied Cryptography and Network Security (ACNS), Kanazawa, Japan. pp. 313–335 (2017). https://doi.org/10.1007/978-3-319-61204-1_16, https://doi.org/10.1007/978-3-319-61204-1_16
44. Yang, R., Li, G., Lau, W.C., Zhang, K., Hu, P.: Model-based security testing: An empirical study on OAuth 2.0 implementations. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS), Xi'an, China. pp. 651–662 (2016). https://doi.org/10.1145/2897845.2897874, https://doi.org/10.1145/2897845.2897874
45. Zhang, Z., Król, M., Sonnino, A., Zhang, L., Rivière, E.: EL PASSO: efficient and lightweight privacy-preserving single sign on. Proc. Priv. Enhancing Technol. **2021**(2), 70–87 (2021)
46. Zhou, Y., Evans, D.: SSOScan: Automated testing of web applications for single sign-on vulnerabilities. In: Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA. pp. 495–510 (2014), https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/zhou