Contribution Title[⋆]

First Author[1][0000−1111−2222−3333], Second Author[2,3][1111−2222−3333−4444], and Third Author[3][2222−−3333−4444−5555]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. The abstract should briefly summarize the contents of the paper in 150–250 words.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

Single sign-on (SSO) systems, such as OpenID Connect [?], OAuth [?] and SAML [?], have been widely deployed as the identity management and authentication infrastructure in the Internet. SSO enables a website, called the *relying party* (RP), to delegate its user authentication to a trusted third party called the *identity provider* (IdP). Thus, a user visits multiple RPs with only a single explicit authentication attempt at the IdP. With the help of SSO, a user no longer needs to remember multiple credentials for different RPs; instead, she maintains only the credential for the IdP, which will generate *identity proofs* for her visits to these RPs. SSO has been widely integrated with many application services. For example, we find that 80% of the Alexa Top-100 websites [?] support SSO, and the analysis on the Alexa Top-1M websites [?] identifies 6.30% with the SSO support. Meanwhile, many email and social network providers (such as Google, Facebook, Twitter, etc.) are serving the IdP roles in the Internet.

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 [?] and Onavo [?]). Hoever, by simply serving the IdP role, these companies can easily collect a large amount of data to reconstruct users' online traces.

User privacy leaks in all existing SSO protocols and implementations. Taking a widely used SSO protocol, OpenID Connect (OIDC), as an example, we explain its login process and the risk of privacy leakage. On receiving a user's

[⋆] Supported by organization x.

login request, the RP constructs a request of identity proof with its identity and redirects it to the IdP. After authenticating the user, the IdP generates an identify proof containing the identities of the user and the RP, which is forwarded to the RP by the user. Finally, the RP verifies the identity proof and allows the user to log in. From such login instances, any 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, if 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.

Imagine that, a user concerning her privacy would avoid to leave her full sensitive information at an application. The user may use multiple web applications and only leave parts of her sensitive messages at each applications, for example, using real name on social website, the address on shopping website and the phone number on Telecom website. And she would try not to leave any linkable message to avoid applications combining her informations, for example, if she leave the email on each applications, they can combine the parts of informations through the 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 your informations based on the SSO account.

Recently more and more IdPs have been considering the user's privacy serious. For example, one of the most worldwide popular instant messaging applications, WeChat, also working as the IdP service provider, enables a user to create different plain accounts for login on multiple RPs. Moreover, Active Directory Federation Services and Oracle Access Management support the use of PPID, the privacy protection scheme suggested in OIDC protocol [?, ?]; identity service providers such as NORDIC APIS and CURITY suggest adopting PPID in SSO to protect user privacy [?, ?].

However, none of the existing techniques proposed by previous research can deal with the privacy concerns comprehensively. Here we give a brief introduction of existing solutions for privacy-preserving SSO and explain the flaws of these schemes.

– *Using different user ID in each RP.* As recommended by NIST [?] and specified in several SSO protocols [?, ?], pairwise pseudonymous identifier (PPID) is generated by the IdP to identify a user to an RP, which cannot be correlated with the user's PPID at another RP. Thus, collusive RPs cannot link a user's logins from her PPIDs. However, PPID-based approaches cannot prevent IdP-based login tracing, since the IdP needs to know which RP the user visits in order to generate the correct identify proof.

– *Simply hiding RP ID from IdP.* For example, SPRESSO [?] 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 ID. However, this type of solution can only provide the same user ID for each RP.
– *Proving user identity based on zero-knowledge proof.* In this type of solution, 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. 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. For security consideration, the s must be too long for user to remember.
– *Completely anonymous SSO system.* Anonymous SSO scheme is proposed to hide the user's identity to both the IdP and RPs with many methods. For example, in the anonymous SSO [?], 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.

As discussed above, none of the existing SSO systems defend against both IdP-based login tracing and RP-based identity linkage, and provided the convenient SSO service on multiple devices at the same time. And, it can not be solved by simply combining existing solutions together. The challenge is that, there is not a simple way for IdP to provide the RP-specific user ID without knowing the RP's identity.

In this paper, we propose XXX, which provides the convenient SSO service with comprehensive protection against bosh IdP-based login tracing and RP-based identity linkage. The key idea of XXX is separating IdP into two parts. The first part is for user authentication and identity proof issuing, which must be completed at server for security consideration. The other part is for RP and user's ID transformation, which must be completed at the user controlled and IdP trusted part to keep RP's identity unknown to IdP. With SGX, the secure hardware supported by intel CPU, the second part can be implemented at user's PC based on the $remoteattestation$, the function provided by SGX. For IdP, it can only achieve a encrypted one-time RP ID, so that the IdP-based login tracing is not impossible. Moreover, for RP, the user controlled IdP part generates the PPID, therefore, it protects user from RP-based identity linkage.

We summarize our contributions as follows.

– We propose the comprehensive solution to hide the users' login traces from both the curious IdP and malicious collusive RPs for convenient SSO system.
– We formally analyze the security of XXX and show that it guarantees the security, while the users' login traces are well protected.
– We have implemented a prototype of XXX, and compare the performance of the XXX prototype with the state-of-the-art SSO systems (i.e., OIDC), and demonstrate its efficiency.

The rest of the paper is organized as follows. We first introduce the background and preliminaries in Section **??**. Then, we describe the identifier-transformation based approach, the threat model, and our UPPRESSO design in Sections **??**, **??** and **??**, followed by a systematical analysis of security and privacy in Section **??**. We provide the implementation specifics and experiment evaluation in Section **??**, discuss the related works in Section **??**, and conclude our work in Section **??**.

2 background

XXX 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

OIDC [**?**] is an extension of OAuth 2.0 to support user authentication, and becomes one of the most prominent SSO authentication protocols. Same as other SSO protocols [**?**], OIDC involves three entities, i.e., *users*, *identity provider (IdP)*, and *relying parties (RPs)*. Both users and RPs have to register at the IdP, the users register at the IdP to create credentials and identifiers (i.e. ID_U), while each RP registers at the IdP with its endpoint information to create its unique identifier (i.e., ID_{RP}) and the corresponding credential. IdP is assumed to securely maintain the attributes of users and RPs. Then, in the SSO authentication sessions, each user is responsible to start a login request at an RP, redirect the messages between RP and IdP, and check the scope of user's attributes provided to the RP; IdP authenticates the user, sets the $PPID$ for the user ID_U at the RP ID_{RP}, constructs the identity proof with $PPID$, ID_{RP} and the user's attributes consented by the user, and finally transmits the identity proof to the RP's registered endpoint (e.g., URL); each RP constructs an identity proof request with its identifier and the requested scope of user's attributes, sends an identity proof request to the IdP through the user, and parses the received identity proof to authenticate and authorize the user. Usually, the redirection and checking at the user are handled by a user-controlled software, called *user agent* (e.g., browser).

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 propose the OIDC implicit flow as the example to illustrate the protocol.

As shown in Figure **??**, the implicit flow of OIDC consists of 7 steps: when a user attempts to log in to an RP (Step 1), the RP constructs a request for identity proof, which is redirected by the user to the corresponding IdP (Step 2). The request contains ID_{RP}, RP's endpoint and a set of requested user attributes. If the user has not been authenticated yet, the IdP performs an authentication process (Step 3). If the RP's endpoint in the request matches the one registered at the IdP, it generates an identity proof (Step 4) and sends it back to the RP (Step

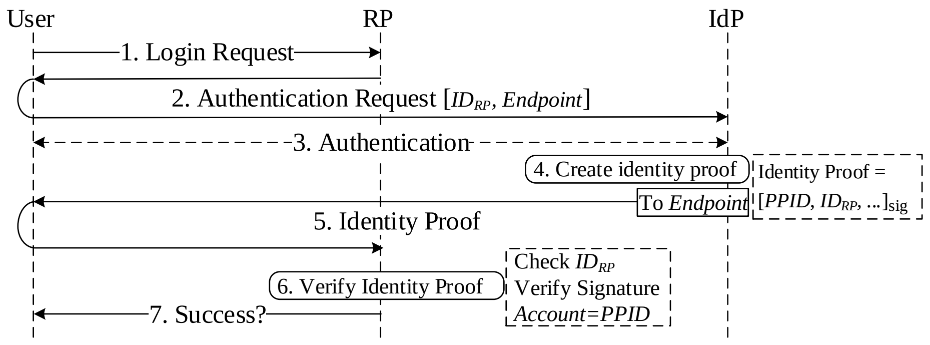

Fig. 1: The implicit protocol flow of OIDC.

5). Otherwise, IdP generates a warning to notify the user about potential identity proof leakage. The RP verifies the id token (Step 6), extracts user identifier from the id token and returns the authentication result to the user (Step 7).

2.2 Intel SGX

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

Remote Attestation. 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

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

3.1 Threat Model

Adversaries' Goal: The adversaries can impersonate an honest user to log in to the honest RP.

Adversaries' Capacities:

– An adversary can act as the malicious user. The adversary can control all the software running outside the enclave, for example, capturing and tempering the message transmission among enclave application, browser and IdP server, decrypting and tempering the https flow outside the enclave, 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 message 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, 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, and never colludes with others (e.g., malicious RPs and users).

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 XXX are assumed to be secure and correctly implemented.

Phishing attack is not considered in this paper.

4 Design of XXX