Understanding the Identity-Transformation Approach in OIDC-Compatible Privacy-Preserving SSO Services

Jingqiang Lin‡, Baitao Zhang‡, Wei Wang‡, Quanwei Cai¶, Jiwu Jing◇, and Huiyang He‡
‡ School of Cyber Security, University of Science and Technology of China
¶ Beijing Zitiao Network Technology Co., Ltd, China
◇ School of Cryptology, University of Chinese Academy of Sciences

Abstract

OpenID Connect (OIDC) enables a user with commercial-off-the-shelf browsers to log into multiple websites, called *relying parties* (RPs), by her username and credential set up in another trusted web system, called the *identity provider* (IdP). *Identity transformations* are proposed in UppreSSO to provide OIDC-compatible SSO services, preventing both IdP-based login tracing and RP-based identity linkage. While security and privacy of SSO services in UppreSSO have been proved, several essential issues of this identity-transformation approach are not well studied. In this paper, we comprehensively investigate the approach as below. Firstly, several suggestions for the efficient integration of identity transformations in OIDC-compatible SSO are explained. Then, we uncover the relationship between identity transformations in SSO and oblivious pseudo-random functions (OPRFs), and present two variations of the properties required for SSO security as well as the privacy requirements, to analyze existing OPRF protocols. Finally, new identity transformations different from those designed in UppreSSO, are constructed based on OPRFs, satisfying different variations of SSO security requirements.

To the best of our knowledge, this is the first time to uncover the relationship between identity transformations in OIDC-compatible privacy-preserving SSO services and OPRFs, and prove the SSO-related properties (i.e., key-identifier freeness, RP designation and user identification) of OPRF protocols, in addition to the basic properties of correctness, obliviousness and pseudo-randomness.

1 Introduction

Single sign-on (SSO) [32, 33, 57] enables a user to visit multiple websites, called *relying parties* (RP), by her username and credential set up in another trusted web system, called the *identity provider* (IdP). OpenID Connect (OIDC) [57] is the most popular SSO protocol in the Internet, for users to access the services by commercial-off-the-shelf (COTS) browsers without a plug-in or extension.

Such SSO services raise the concerns on user privacy, because the original designs facilitate interested parties to track a user. For instance, when requesting identity tokens from an IdP to visit an RP, an OIDC user submits the target RP's identity, so that the curious IdP could track the user's all login activities [22, 23, 29], called *IdP-based login tracing*. Meanwhile, if a user visits different RPs with tokens binding an identical user identity, colluding RPs could link the accounts across these RPs to profile this user [27, 29, 50], called *RP-based identity linkage*.

Several solutions provide OIDC-compatible SSO services [22, 23, 27–31, 43, 61] for a user with COTS browsers, while preventing privacy leakage (i.e., IdP-based login tracing, RP-based identity linkage, or both). However, none of them except UppreSSO [29, 30] offer all describes properties. Some of these schemes prevent only one privacy threat [22, 23, 27, 31, 43], and the others introduce extra trusted servers (or components) in addition to the honest-but-curious IdP [28, 61] or impractical computation costs [31]. Compared with them, UppreSSO proposes an identity-transformation approach to protect users against these two privacy risks, without trusted servers more than an honest-but-curious IdP. When a user visits an RP in UppreSSO, *ephemeral pseudo-identities* are generated for the visited RP and the user, and signed in an identity token, which enables the user to log into the target RP as her *permanent account* at this RP.

While the identity-transformation approach has been described, proved, implemented, and evaluated in the implicit flow of OIDC [29, 30], and integrated in the authorization code flow [35], there are still essential issues not well studied. Firstly, the integration of identity transformations in OIDC-compatible SSO sometimes introduces unnecessary calculations [29, 35] or extra trusted servers [35]. In this paper we present several suggestions for the more efficient integration. Moreover, the suggestion for an RP to accept identity tokens efficiently, inspires us to analyze variations of the properties required for secure SSO, i.e., *user identification* and *RP designation*, either *with* or *without* checking the RP's pseudo-identity in signed tokens. Secondly, we construct more identity transformations qualified for OIDC-compatible privacy-preserving SSO. The relationship between the identity-transformation approach and oblivious pseudo-random functions (OPRFs) is uncovered. This relationship then helps us to analyze existing OPRF protocols, following the variations of user identification and RP designation as well as the properties required for privacy. Finally, based on the analysis results, new identity transformations are constructed for OIDC-compatible privacy-preserving SSO, satisfying different variations of security requirements.

To the best of our knowledge, this is the first time to (*a*) uncover the relationship between identity transformations in OIDC-compatible privacy-preserving SSO services and OPRFs, and (*b*) prove the SSO-related properties of OPRFs (i.e., key-identifier freeness, RP designation and user identification), which are independent of the basic OPRF properties of correctness, obliviousness and pseudo-randomness.

The rest of this paper is organized as follows. Section 2 briefly describes the identity-transformation approach. Section 3 compares existing privacy-preserving SSO schemes and presents suggestions for the efficient integration of identity transformations in SSO systems. We construct and prove new identity transformations in Section 4, and discuss related work in Section 5. Section 6 concludes this paper.

2 Identity Transformations in UppreSSO

This section describes the identity-transformation approach proposed in UppreSSO [29, 30], implementing privacy-preserving SSO services accessed from COTS browsers. Only the fundamental designs of this approach are included in this section, and more details can be found in [30].

2.1 System Model and Initialization

An OIDC-compatible SSO system consists of several RPs, a set of users, and an *honest-but-curious* IdP signing identity tokens for a user to visit these RPs. *Malicious* adversaries could fully control some RPs and users, attempting to break the security or privacy guarantees for honest users and RPs.

Authenticated and confidential links are established between honest entities, and the cryptographic primitives are secure. The software stack of an honest entity is correctly implemented to deliver messages to receivers as expected.

$\mathbb{E}$ is an elliptic curve over a finite field $\mathbb{F}_q$. G is a generator on $\mathbb{E}$, and the order of G is a prime n. Each user registers at the IdP with a unique identity, denoted as $ID_U = u \in \mathbb{Z}_n$, and each RP is assigned a unique identity $ID_{RP} = [r]G$, where $[r]G$ denotes the addition of G on the elliptic curve r times.

During the registrations u and r are randomly selected in $\mathbb{Z}_n$ by the IdP, and kept *unknown* to users and RPs. ID_U is processed only by the IdP internally and never enclosed in any messages, and r is not used any more after the registration.

Finally, the user with $ID_U = u$ is automatically assigned an account $Acct = \mathcal{F}_{Acct*}(ID_U, ID_{RP}) = [ID_U]ID_{RP} = [ur]G$ at the RP with $ID_{RP} = [r]G$.

2.2 The Implicit Flow of OIDC with Identity Transformations

In addition to $\mathcal{F}_{Acct*}()$, the following functions are defined [29, 30]:

- $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t) = [t]ID_{RP} = [tr]G$, where t is a random number in $\mathbb{Z}_n$.
- $PID_U = \mathcal{F}_{PID_U}(ID_U, PID_{RP}) = [ID_U]PID_{RP} = [utr]G$.
- $Acct = \mathcal{F}_{Acct}(PID_U, t) = [t^{-1}]PID_U = [t^{-1}utr]G = [ur]G = \mathcal{F}_{Acct*}(ID_U, ID_{RP})$.

$\mathcal{F}_{Acct*}()$ determines a user's account unique at every RP, while $\mathcal{F}_{PID_{RP}}()$ and $\mathcal{F}_{PID_U}()$ transform an RP's identity and a user's identity into two *pseudo-identities* in a login to prevent IdP-based login tracing and RP-based identity linkage, respectively. Finally, $\mathcal{F}_{Acct}()$ enables an RP to derive a user's *permanent* account based on *ephemeral* PID_U, equal to the one determined by $\mathcal{F}_{Acct*}()$.

The implicit flow of OIDC with identity transformations works as follows.

1. When attempting to access protected resources at an RP, a user prepares her user agent by downloading scripts. The trusted part of user-agent scripts is downloaded from the honest-but-curious IdP, and the other script is downloaded from the visited RP to forward identity tokens.
2. The user obtains ID_{RP} of the target RP, randomly selects $t \in \mathbb{Z}_n$, and calculates $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t)$. Then, she requests an identity token for PID_{RP} from the IdP, and t is sent to the RP.
3. The IdP authenticates the user, if not authenticated yet. It then calculates $PID_U = \mathcal{F}_{PID_U}(ID_U, PID_{RP})$, signs an identity token binding PID_{RP} and PID_U, and returns this token to the user.
4. The user forwards the signed token to the visited RP. The RP verifies the received token, extracts PID_U from it, calculates $Acct = \mathcal{F}_{Acct}(PID_U, t)$, and allows the user to log in as $Acct$.

We particularly note some issues [29, 30]. In UppreSSO user operations are implemented on a COTS browser with two scripts (or windows), so it works compatibly in OIDC services with pop-up UX [26, 46, 59], but not redirect UX. In these operations, the trapdoor t is kept *secret* to the honest-but-curious IdP, and known to only the user and the visited RP. The target RP is designated, as PID_{RP} is calculated based on ID_{RP} of the visited RP (but not any other RP). This target is ensured by a signed RP certificate binding ID_{RP} and the RP's endpoint to receive identity tokens. The RP certificate is verified in browsers, and the token is forwarded to only the endpoint specified in this certificate.

After verifying the IdP's signature on a received token, an RP does *not* check whether PID_{RP} in this token equals to $[t]ID_{RP}$ or not. This efficient design does not result in attacks, because an *unmatching* token results in a *meaningless* account corresponding to a *non-existing* user, which has been proved [30].

2.3 Security and Privacy of SSO Services

This section briefly explains the properties related to security and privacy, which have been proved [30].

2.3.1 Security. As authenticity, confidentiality and integrity of identity tokens in SSO services are ensured by secure communications (i.e., HTTPS among entities and security mechanisms of COTS browsers [26, 29, 30, 46]) and digital signatures (i.e., signed identity tokens and RP certificates [29, 30]), the following sufficient conditions of *secure* SSO services are proved with the identity transformations [30], namely *user identification* and *RP designation*, under the adversarial model including (*a*) an honest-but-curious IdP and (*b*) malicious RPs colluding with users.

In UppreSSO an identity token, denoted as TK, is signed by the IdP to bind $PID_{RP} = [t]ID_{RP}$ and $PID_U = [ID_U]PID_{RP}$.

User Identification: *At the designated RP, TK identifies only the user who requests this token from the IdP. That is, based on TK, the designated honest RP will derive only the account corresponding to the user requesting TK.*

RP Designation: *TK designates the target RP, and based on TK only the designated (honest) RP derives meaningful accounts corresponding to registered users. That is, at any honest RPs other than the designated one, no meaningful account is derived based on TK.*

As described in Section 2.2, provided that a user selects t, calculates $PID_{RP} = [t]ID_{RP}$, and sends t to the target RP, once RP designation is ensured as above, an (honest) RP does *not* need to check whether PID_{RP} enclosed in the token equals to $[t]ID_{RP}$ because no meaningful account will be derived at any honest RPs other than the designated one.

However, if an RP selects t, calculates $PID_{RP} = [t]ID_{RP}$, and sends PID_{RP} to users, impersonation attacks happen as below, even when an (honest) RP checks PID_{RP} in received tokens. For example, a malicious user receives PID_{RP} when visiting an honest RP. This

Table 1: OIDC-compatible Privacy-preserving SSO Solutions

	OIDC w/ PPID	Miso	UP-SSO	BrowserID	Spresso	Poidc/AIF	UppreSSO
IdP-based Login Tracing	⊥	√	√	√	√	√	√
RP-based Identity Linkage	√	√	√	⊥	⊥	⊥[3]	√
Extra Trusted Server	√	⊥	⊥[1]	√	⊥[2]	√	√

√ means a privacy threat is prevented or extra trusted servers more than an honest IdP are eliminated, and ⊥ means not.

1. Although UP-SSO [28] does not introduce an independent trusted server, it requires a fully-trusted component (i.e., Intel SGX enclave) on the user side.
2. In Spresso [23] an extra trusted Forwarder distributes user-agent scripts to users, because a malicious IdP is assumed and then scripts downloaded from this IdP are potentially malicious (but in other schemes an honest-but-curious IdP is assumed).
3. A variation of Poidc [31] proposes to also hide a user's identity in the commitment and prove this to the IdP in zero-knowledge, so RP-based identity linkage is also prevented *theoretically*; but it takes seconds to generate such a zero-knowledge proof even on a powerful server [18, 48], which is *impracticable* for an SSO user agent.

malicious user does not request identity tokens for PID_{RP} from the IdP, but forwards PID_{RP} to some colluding RP. Once an honest user visits this colluding RP, it immediately sends PID_{RP} to this honest victim to obtain an identity token, shared among colluding adversaries. Finally, the malicious user exploits this token binding PID_{RP}, to successfully log in as the honest victim's account at the honest RP. In this attack, the honest RP is *not* the designated one, because PID_{RP} is not calculated based on its ID_{RP}.

A naive prevention against such impersonations in the above procedure is to *additionally* check $PID_{RP} = [t]ID_{RP}$ by the user, who receives PID_{RP} and also t from the visited RP. Thus, PID_{RP} is calculated repeatedly: The RP selects t, calculates $PID_{RP} = [t]ID_{RP}$, and sends them to the user, who checks whether PID_{RP} equals to $[t]ID_{RP}$ by calculating $[t]ID_{RP}$ again.[1]

The calculation of $PID_{RP} = [t]ID_{RP}$ *only* by users [30], prevents such impersonation attacks effectively and efficiently. This results in no attacks, because (*a*) RP designation is ensured as above with the proposed identity transformations, which has been proved [30], and (*b*) as $PID_{RP} = [t]ID_{RP}$ is calculated by the user requesting identity tokens from the IdP, the RP with ID_{RP} is designated accordingly.

2.3.2 Privacy. The identity transformations provide *IdP untraceability* against IdP-based login tracing by the honest-but-curious IdP, and *RP unlinkability* against RP-based identity linkage by colluding RPs [29, 30].

Among the messages received by an honest-but-curious IdP in UppreSSO, only PID_{RP} is calculated based on ID_{RP}, while RP-based identity linkage is launched by malicious RPs colluding with some users. So the following privacy-related properties are defined.

IdP Untraceability: *The IdP cannot distinguish PID_{RP} from a uniformly random variable. That is, the IdP learns nothing on the visited RP from PID_{RP}.*

RP Unlinkability: *Malicious RPs colluding with users, cannot link any login initiated by an honest user visiting a malicious RP, to any subset of logins visiting any other colluding RPs by honest users. That is, logins visiting different malicious RPs by any honest user, are indistinguishable to these colluding RPs.*

[1]This inefficient design was originally presented in the preliminary version of UppreSSO [29] in 2022, and then inherited by ArpSSO [35] in 2024. It is improved later [30], and this unnecessary repeated calculation of PID_{RP} is finally eliminated in the integration of identity transformations.

3 OIDC-Compatible Privacy-Preserving SSO

We compare existing SSO solutions with privacy protections, and investigate the integration of identity transformations in OIDC.

3.1 Comparison

Table 1 compares existing privacy-preserving SSO solutions. We do not consider identity federation [2, 37, 38, 49, 54, 62] which requires a browser plug-in or extension.

Pairwise pseudonymous identifiers (PPIDs) [27] protect user privacy against colluding RPs. An IdP assigns a unique PPID for a user at every RP and encloses it in identity tokens, so that colluding RPs cannot link accounts across these RPs. IdP-based login tracing still exists because the IdP needs the visited RP's identity to set PPIDs. UP-SSO [28] runs a trusted Intel SGX enclave on the user side, which is remotely attested by the IdP and then receives a secret to generate PPIDs for a user. Miso [61] decouples the calculation of PPIDs from an IdP, and introduces a fully-trusted server based on Intel SGX, called Mixer, to calculate PPIDs based on ID_U, ID_{RP} and a secret, so that it prevents also IdP-based login tracing for ID_{RP} is disclosed to the Mixer but not the IdP. However, in Miso the Mixer could track a user's all login activities.

Some schemes prevent IdP-based login tracing but are vulnerable to RP-based identity linkage, due to unique user identities in identity tokens. In BrowserID [22] an IdP issues a "user certificate" binding a user identity to an ephemeral key pair. The user then uses the corresponding private key to sign an "identity assertion" binding the target RP's identity, and sends both of them to the RP. In Spresso an RP creates a one-time tag (or pseudo-identity) for each login [23], while in Poidc [31] or AIF [43] a user requests an identity token by sending a hash commitment on the target RP's identity, which are bound in the token with the user's unique identity.

Compared with other schemes, the identity-transformation approach proposed in UppreSSO [29, 30] prevents both IdP-based login tracing and RP-based identity linkage, and requires no trusted server more than the honest-but-curious IdP. The experimental performance evaluation with COTS browsers demonstrated its reasonable overheads. Besides, ArpSSO shifted the identity transformations on elliptic curves [29] into a finite field $\mathbb{F}_q$, and integrated them in the authorization code flow of OIDC. Thus, IdP-based login tracing and RP-based identity linkage are prevented in ArpSSO by

the integrated identity transformations.[2] However, it introduces extra trusted servers to distribute and verify the user-agent scripts, which are unnecessary due to the common honest-but-curious assumptions of the IdP in ArpSSO.

3.2 Integrating Identity Transformations in OIDC Services

While unnecessity of PID_{RP} checking by an RP is proved in the implicit flow [30] (see Section 2.3.1), it is applicable to the integration of identity transformations in the authorization code flow. We explain this applicability by improving ArpSSO [35] as below.

ArpSSO shifts the identity transformations into a finite field, where $ID_U = u$, $ID_{RP} = g^r$, $PID_{RP} = ID_{RP}^{t_1} = g^{rt_1}$ and $PID_U = PID_{RP}^u = g^{rut_1}$ in $\mathbb{F}_q$. Meanwhile, it employs PS signing [55] to verify that PID_{RP} is calculated based on the visited RP's identity: During the registration an RP obtains a PS signature $(g^r, g^{r(x+y_1\mathcal{D})})$ for its domain $\mathcal{D}$, where (x, y_1) is the IdP's PS signing key.

Let $\sigma_2 = g^{r(x+y_1\mathcal{D})}$, and the login flow in ArpSSO is described as below. It is worth nothing that, t_1 and t_2 in this description, correspond to k and t in the original expressions in [35], respectively. The second random number t_2, is generated for RP anonymous authentication [62] to the IdP in the retrieval of tokens in the authorization code flow, unrelated to the identity transformations.

1. When visited by a user, an RP randomly selects $t_1, t_2 \in \mathbb{F}_q$, and calculates $(g^{rt_1}, (\sigma_2 g^{rt_2})^{t_1})$ which is sent to the user along with t_2. Moreover, $(g^{rt_1}, (\sigma_2 g^{rt_2})^{t_1})$ is kept by the RP for the subsequent authentication to the IdP.
3. The user calculates $v = (1/g^{rt_1})^{t_2}$ and $w = v(\sigma_2 g^{rt_2})^{t_1}$, and verifies whether (g^{rt_1}, w) is a valid PS signature for $\mathcal{D}$, which is the visited domain.
4. The user requests an identity token for $PID_{RP} = g^{rt_1}$, and receives an authorization code. This authorization code is forwarded to the RP, and used to retrieve the token binding $PID_{RP} = g^{rt_1}$ and $PID_U = PID_{RP}^u = g^{urt_1}$. The RP finally allows the user to log in as $Acct = (g^{urt_1})^{1/t_1} = g^{ur}$.

In the above procedure, $PID_{RP} = ID_{RP}^{t_1} = g^{rt_1}$ is first calculated by the visited RP and later checked actually by the user calculating $v = (1/g^{rt_1})^{t_2}$ and $w = v(\sigma_2 g^{rt_2})^{t_1}$. We improve it to eliminate the unnecessary PID_{RP} checking by an RP as follows.

1. The RP directly sends (g^r, σ_2) to the user.
2. After verifying that (g^r, σ_2) is a valid PS signature for $\mathcal{D}$, the user randomly selects t_1, calculates g^{rt_1}, and sends both g^{rt_1} and t_1 to the RP.
3. The RP randomly selects t_2, and calculates only $(\sigma_2 g^{rt_2})^{t_1}$ to construct $(g^{rt_1}, (\sigma_2 g^{rt_2})^{t_1})$ for the subsequent anonymous authentication to the IdP.
4. An identity token binding $PID_{RP} = g^{rt_1}$ and $PID_U = g^{urt_1}$ is requested by the user and signed by the IdP. This token allows the user to log in.

Thus, $PID_{RP} = g^{rt_1}$ is calculated only by the user. We eliminate the calculations of $v = (1/g^{rt_1})^{t_2}$ and $w = v(\sigma_2 g^{rt_2})^{t_1}$, while the security and privacy guarantees are still strictly provided [30].

[2] In ArpSSO [35] the identity transformations of ID_{RP}-PID_{RP} and ID_U-PID_U are called *RP anonymization* and *user identity mix-up*, respectively.

Next, to efficiently integrate identity transformations in the authorization code flow, in the retrieval of identity tokens, we recommend (*a*) the widely-used OIDC option of proof key for code exchange (PKCE) [56], and (*b*) anonymous credentials such as ring signature [5] and privacy-pass token [14, 60], to serve for RP anonymous authentication to the IdP. In every login an RP generates a PKCE code verifier and hashes it to a challenge. This PKCE challenge is sent to the user, and forwarded to the IdP. Then, to retrieve a token from the IdP, the RP submits an authorization code and the verifier over anonymous networks [16, 58], both of which are verified by the IdP, after anonymously authenticated to the IdP (i.e., along with a ring signature or privacy-pass token). Then, the target RP anonymously retrieves the token, and any other registered RP cannot obtain the token from the IdP, even if it intercepts the authorization code. Ring signature or privacy-pass token is much more efficient than the zero-knowledge proofs [62] adopted in ArpSSO.

Alternative to RP certificates [29, 30] and PS signatures [55] sent from RPs, hashing-to-elliptic-curves [20] is proposed to establish the relationship between ID_{RP} and the visited RP [29, 30] with no cost of messages transmitted. However, the design of RP certificates can be utilized without extra messages transmitted: ID_{RP} is enclosed in an RP's HTTPS certificate, signed by a trusted certification authority (CA) and used in secure communications between the RP and users. Then, the user-agent scripts directly obtain this certificate [44] and then ID_{RP}. This method does not introduce extra trusted entities, because HTTPS is adopted in SSO services and the CA has been implicitly trusted by all users.

The above suggestions help to efficiently integrate identity transformations in OIDC services, while some have been applied or discussed but not clearly explained.

4 Constructing Identity Transformations Based on OPRFs

4.1 Basic Properties of OPRFs

Given a pseudo-random function $z = \mathcal{PR}(k, x)$, where k is the *secret key* held by an OPRF server and x is a *private input* from OPRF users, the OPRF server and users cooperate as below [10, 24, 34, 39, 42, 51]:

1. The server sends a parameter ω. This step is optional.
2. An OPRF user blinds her input x into $x' = \mathcal{BL}(x, t, \omega)$ using a random number t, and sends x' to the OPRF server.
3. The OPRF server calculates the output as $z' = \mathcal{OPR}(k, x')$.
4. On receiving z', the user unblinds it into $z = \mathcal{UBL}(z', t, \omega)$.

This formalization covers typical protocols, as shown in Table 2: (*a*) HashDH [24, 34], (*b*) the NR OPRF using homomorphic encryption (HE) [1, 39], denoted as NR_{HE} in this paper, (*c*) the DY OPRF using HE [6, 42, 51], denoted as DY_{HE}, and (*d*) 2HashRSA [39].

An OPRF protocol satisfies the basic properties [10, 24]:

- **Correctness:** For any x, k, t, and ω, $\mathcal{UBL}(z', t, \omega)$ is equal to $\mathcal{PR}(k, x)$.
- **Pseudo-Randomness:** In an OPRF user's view, z is indistinguishable from uniformly random variables, and she learns nothing on k.
- **Obliviousness:** The OPRF server learns nothing on x.

Table 2: Functions and Parameters of Typical OPRFs

	HashDH	NR_{HE}	DY_{HE}	2HashRSA
$z = \mathcal{PR}(k, x)$	$k \xleftarrow{\$} \mathbb{F}_q$ $x \xleftarrow{\$} \mathbb{F}_q$ $z = x^k$	$k = (a_0, a_1, \cdots, a_l) \xleftarrow{\$} \mathbb{F}_q^{l+1}$ $x = \tilde{x}_1 \cdots \tilde{x}_l \in \{0,1\}^l$ $z = g^{a_0} \prod_{i=1}^{l} a_i^{\tilde{x}_i}$	$k \xleftarrow{\$} \mathbb{F}_q$ $x \xleftarrow{\$} \mathbb{F}_q$ $z = g^{1/(k+x)}$	$(N, e, k) \Leftarrow RSA()$ $x \xleftarrow{\$} \mathbb{Z}_N$ $z = H_2(x, x^k)$
ω	-	$(r_1, \cdots, r_n) \xleftarrow{\$} \mathbb{F}_q^l$ $\omega = g^{a_0} \prod_{i=1}^{l} 1/r_i$	$(sk, pk) \Leftarrow HE()$ $\omega = Enc(k)$	$\omega = (N, e)$
$x' = \mathcal{BL}(x, t, \omega)$	$t \xleftarrow{\$} \mathbb{F}_q$ $x' = x^t$	$t = (sk, pk) \Leftarrow HE()$ $m_i = (1 - \tilde{x}_i, \tilde{x}_i)$ $x' = \{Enc(m_{i=1,\cdots,l})\}$	$t \xleftarrow{\$} \mathbb{F}_q$ $x' = (\omega Enc(x))^t$	$t \xleftarrow{\$} \mathbb{Z}_N$ $x' = xt^e$
$z' = \mathcal{OPR}(k, x')$	$z' = x'^k$	$z' = \{Enc(m_i)^{(r_i, r_i a_i)}\}$	$t(k + x) = Dec(x')$ $z' = g^{1/t(k+x)}$	$z' = x'^k$
$z = \mathcal{UBL}(z', t, \omega)$	$z = z'^{1/t}$	$\{r_i a_i^{\tilde{x}_i}\} = Dec(z')$ $z = \omega \prod_{i=1}^{l} r_i a_i^{\tilde{x}_i}$	$z = z'^t$	$z = H_2(x, z'/t)$

1. In the original HashDH, $z = H_1(x)^k$ and $H_1()$ is a collision-free hash function outputting uniformly random elements in $\mathbb{F}_q$. As x is randomly selected in $\mathbb{F}_q$, we set $H_1(x) = x$. Besides, it is easy to shift HashDH to elliptic curves, resulting in HashECDH. The conclusions in this paper are applicable to HashECDH.
2. $(sk, pk) \Leftarrow HE()$ generates a key pair of an additively HE scheme such as Paillier [53]. $Enc()$ and $Dec()$ represents its encryption and decryption, respectively. So $Enc(a)Enc(b) = Enc(a + b)$ and $Enc(a_1, a_2)^{(b_1, b_2)} = Enc(a_1)^{b_1} Enc(a_2)^{b_2} = Enc(a_1 b_1 + a_2 b_2)$.
3. $(N, e, k) \Leftarrow RSA()$ generates an RSA key pair, and $H_2()$ is a collision-free hash function outputting uniformly random elements in $\mathbb{Z}_N$.
4. In 2HashRSA the calculations are conducted in $\mathbb{Z}_N$, while those of other OPRFs are done in $\mathbb{F}_q$.

4.2 Building SSO Services Based on OPRFs

The ($\mathbb{F}_q$ versions of the) identity transformations proposed in UppreSSO [29, 30] *mathematically* utilize the same functions as the HashDH OPRF protocol [39, 52]; that is, four functions of HashDH (i.e., $\mathcal{PR}()$, $\mathcal{BL}()$, $\mathcal{OPR}()$, and $\mathcal{UBL}()$), actually work as $\mathcal{F}_{Acct*}()$, $\mathcal{F}_{PID_{RP}}()$, $\mathcal{F}_{ID_U}()$, and $\mathcal{F}_{Acct}()$ in UppreSSO, respectively.

However, not every OPRF is ready to work as the identity transformations in privacy-preserving SSO. More properties of these functions of an OPRF protocol are required [30] to work as the identity transformations in privacy-preserving SSO, and these extended properties (see Sections 4.3 and 4.4 for details) have not been investigated in the literature [10].

Next, we construct identity transformations qualified for privacy-preserving SSO, based on OPRF protocols other than HashDH. This construction explains the extended properties well.

Given an OPRF protocol conforming to the formalization in Section 4.1, SSO services are built as below. During the registrations, $ID_U = k$ and $ID_{RP} = x$ are assigned to a user and an RP, respectively. $Acct = \mathcal{F}_{Acct*}(ID_U, ID_{RP}) = \mathcal{PR}(k, x) = z$ is automatically assigned to a user at each RP. $ID_{RP} = x$ is publicly-known, while $ID_U = k$ is kept *secret* and known to only the IdP; otherwise, RP unlinkability is broken [30]: Colluding RPs could calculate $Acct = \mathcal{F}_{Acct*}(ID_U, ID_{RP}) = \mathcal{PR}(k, x)$ for each known ID_U at these RPs and accordingly link the accounts.

Table 3 lists the corresponding variables and functions. Note that ω sometimes depends on k, but not disclosing any information on k, and it plays different roles in these protocols: (*a*) ω does not exist in HashDH; (*b*) in NR_{HE} $\omega = g^{a_0} \prod_{i=1}^{n} 1/r_i$ is an argument of $\mathcal{UBL}()$, (*c*) in DY^{HE} $\omega = Enc(k)$ is processed only in $\mathcal{BL}()$; and (*d*) in 2HashRSA, $\omega = (N, e)$ is processed in both $\mathcal{BL}()$ and $\mathcal{UBL}()$.

The IdP, RPs, and users follow these specifications in a login.

1. The IdP sends ω to a user, who is visiting an RP identified as $ID_{RP} = x$, if needed.
2. The user requests an identity token for PID_{RP} from the IdP, after randomly selecting t and calculating $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t, \omega) = \mathcal{BL}(x, t, \omega) = x'$. Meanwhile, it sends t to the RP (as well as ω, if ω is an argument of $\mathcal{UBL}()$).
3. After authenticating the user as k, the IdP calculates $PID_U = \mathcal{F}_{PID_U}(ID_U, PID_{RP}) = \mathcal{OPR}(k, x') = z'$, and signs TK binding PID_{RP} and PID_U (i.e., x' and z').
4. TK is forwarded by the user to the RP in the implicit flow of OIDC, or in the authorization code flow it is retrieved by the RP after anonymously authenticated to the IdP.
5. Base on TK, the RP derives $Acct = \mathcal{F}_{Acct}(PID_U, t, \omega) = \mathcal{UBL}(z', t, \omega) = z$.

4.3 SSO-Related Properties of OPRFs

First of all, in an SSO system built as above, *correctness* of OPRFs, i.e., $\mathcal{UBL}(z', t, \omega) = \mathcal{PR}(k, x)$, ensures that $\mathcal{F}_{Acct}(PID_U, t, \omega) = \mathcal{F}_{Acct*}(ID_U, ID_{RP})$, i.e., correct accounts are always derived at honest RPs in the case of no attack.

In such a *secure* SSO system, an identity token TK requested by a user to visit an RP, enables only this user to log into only this honest target RP as her account at this RP. TK binding $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t)$ and $PID_U = \mathcal{F}_{PID_U}(ID_U, PID_{RP})$, (*a*) designates *only* the RP with ID_{RP} and (*b*) identifies *only* the user with ID_U at this designated RP. This produces two properties for SSO security, *user identification* and *RP designation*.

An RP may derive accounts based on (*a*) *any* signed identity tokens, as demonstrated in UppreSSO [30] and explained in Section

Table 3: Identity Transformations in SSO vs. OPRFs

	Identity Transformation in SSO	OPRF
Variable	ID_U	k
	ID_{RP}	x
Function	$Acct = \mathcal{F}_{Acct*}(ID_U, ID_{RP})$	$z = \mathcal{PR}(k, x)$
	$PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t, \omega)$	$x' = \mathcal{BL}(x, t, \omega)$
	$PID_U = \mathcal{F}_{PID_U}(ID_U, PID_{RP})$	$z' = \mathcal{OPR}(k, x')$
	$Acct = \mathcal{F}_{Acct}(PID_U, t, \omega)$	$z = \mathcal{UBL}(z', t, \omega)$

3.2, or (*b*) *only* tokens binding *matching* PID_{RP}, as implied by the original OIDC protocol [57] and other privacy-preserving SSO solutions [22, 23, 27, 28, 31, 43, 61] where only tokens binding matching RP (pseudo-)identities are accepted. These different operations require two *alternative* definitions of RP designation: *RP designation w/o PID_{RP} checking* and *RP designation w/ PID_{RP} checking*.

That is, if RP designation w/o PID_{RP} checking is ensured, an RP accepts *any* signed tokens to derive accounts; alternatively, if RP designation w/ PID_{RP} checking is ensured, an RP accepts *only* signed tokens binding *matching* PID_{RP}.

We analyze these two definitions (or variations) of RP designation, and the corresponding definitions of user identification. Secure privacy-preserving SSO services can be built based on an OPRF with any variations of these properties, if the corresponding RP operations are specified. Anyway, the more efficient protocol operations are recommended, when stronger properties (i.e., user identification and RP designation w/o PID_{RP} checking) are ensured.

Let $\mathbb{ID}_{RP} = \{ID_{RP_{j=1,\cdots,p}}\}$, $\mathbb{ID}_U = \{ID_{U_{i=1,\ldots,s}}\}$, $\mathbb{x} = \{x_1, \cdots, x_p\}$ and $\mathbb{k} = \{k_1, \cdots, k_s\}$. Therefore, $\mathbb{A}cct = \{Acct_{i,j=1,\cdots,p}|_{i=1,\cdots,s}\} = \{\mathcal{F}_{Acct*}(ID_{U_i}, ID_{RP_j})\}$ and $\mathbb{z} = \{z_{i,j=1,\cdots,p}|_{i=1,\cdots,s}\} = \{\mathcal{PR}(k_i, x_j)\}$.

4.3.1 Recommended security properties. Firstly, the recommended security properties w/o PID_{RP} checking are analyzed.

User Identification w/o PID_{RP} Checking: Based on TK, the designated honest RP derives *only* the account corresponding to the user requesting TK and not any other meaningful accounts. For any ID_{RP}, (*a*) if $ID_{\hat{U}} \neq ID_{\check{U}}$, $\mathcal{F}_{Acct*}(ID_{\hat{U}}, ID_{RP}) \neq \mathcal{F}_{Acct*}(ID_{\check{U}}, ID_{RP})$ as we have $\mathcal{F}_{Acct}(PID_U, t, \omega) = \mathcal{F}_{Acct*}(ID_U, ID_{RP})$, and (*b*) given known $\mathbb{ID}_{RP}$, known $\mathbb{A}cct$, unknown $\mathbb{ID}_U$, and any TK binding $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, \hat{t}, \hat{\omega})$ and $PID_{\hat{U}} = \mathcal{F}_{PID_U}(ID_{\hat{U}}, PID_{RP})$, malicious RPs and users cannot find $\check{t}$ and $\check{\omega}$ which satisfy that $\mathcal{F}_{Acct}(PID_{\hat{U}}, \check{t}, \check{\omega}) = \mathcal{F}_{Acct*}(ID_{\check{U}}, ID_{RP})$, where $ID_{\check{U}} \neq ID_{\hat{U}}$ and $ID_{\check{U}}, ID_{\hat{U}} \in \mathbb{ID}_U$.

We list the corresponding requirement for OPRFs: For any x, (*a*) if $\hat{k} \neq \check{k}$, then $\mathcal{PR}(\hat{k}, x) \neq \mathcal{PR}(\check{k}, x)$, and (*b*) given known $\mathbb{x}$, known $\mathbb{z}$, unknown $\mathbb{k}$, and any protocol instance $(x, \hat{x}', \hat{z}', \hat{t}, \hat{\omega}, \hat{z})$ generated with unknown $\hat{k}$, malicious OPRF users cannot find $\check{t}$ and $\check{\omega}$ satisfying that $\mathcal{UBL}(\hat{z}', \check{t}, \check{\omega}) = \mathcal{PR}(\check{k}, x)$, where $\check{k} \neq \hat{k}$ and $\check{k}, \hat{k} \in \mathbb{k}$.

RP Designation w/o PID_{RP} Checking: At any honest RPs other than the designated one, based on TK no meaningful account is derived. That is, given known $\mathbb{ID}_{RP}$, known $\mathbb{A}cct$ and unknown $\mathbb{ID}_U$, malicious adversaries cannot find j_1, j_2, i_1, i_2, t_1, t_2, ω_1 and ω_2 satisfying that $\mathcal{F}_{Acct}(\mathcal{F}_{PID_U}(ID_{U_{i_1}}, \mathcal{F}_{PID_{RP}}(ID_{RP_{j_1}}, t_1, \omega_1)), t_2, \omega_2) = \mathcal{F}_{Acct*}(ID_{U_{i_2}}, ID_{RP_{j_2}})$ where $j_1 \neq j_2$, $1 \leq j_1, j_2 \leq p$, and $1 \leq i_1, i_2 \leq s$.

This requirement is proposed for OPRFs: Given known $\mathbb{x}$, known $\mathbb{z}$, and unknown $\mathbb{k}$, malicious users cannot find j_1, j_2, i_1, i_2, t_1, t_2, ω_1 and ω_2 satisfying that $\mathcal{UBL}(\mathcal{OPR}(k_{i_1}, \mathcal{BL}(x_{j_1}, t_1, \omega_1)), t_2, \omega_2) = \mathcal{PR}(k_{i_2}, x_{j_2})$, where $j_1 \neq j_2$, $1 \leq j_1, j_2 \leq p$ and $1 \leq i_1, i_2 \leq s$.

4.3.2 Alternative security properties. The alternative security properties w/ PID_{RP} checking in SSO services are defined as below. Note that t and PID_{RP} are checked *simultaneously* (as well as ω, if ω is an argument of $\mathcal{BL}()$), when $PID_{RP} = \mathcal{F}_{PID_{RP}}(ID_{RP}, t, \omega) = \mathcal{BL}(x, t, \omega)$ is checked by the RP which receives a signed identity token binding PID_{RP}.

User Identification w/ PID_{RP} Checking: Based on TK, the designated honest RP derives *only* the account owned by the user requesting TK. For any ID_{RP}, (*a*) if $ID_{\hat{U}} \neq ID_{\check{U}}$, then $\mathcal{F}_{Acct*}(ID_{\hat{U}}, ID_{RP}) \neq \mathcal{F}_{Acct*}(ID_{\check{U}}, ID_{RP})$, and (*b*) malicious adversaries cannot find t_1, t_2, ω_1 and ω_2 satisfying that $\mathcal{F}_{PID_{RP}}(ID_{RP}, t_1, \omega_1) = \mathcal{F}_{PID_{RP}}(ID_{RP}, t_2, \omega_2)$ where $(t_1, \omega_1) \neq (t_2, \omega_2)$.

The property for OPRFs is listed accordingly: For any given x, (*a*) if $\hat{k} \neq \check{k}$, then $\mathcal{PR}(\hat{k}, x) \neq \mathcal{PR}(\check{k}, x)$, and (*b*) malicious OPRF users cannot find t_1, t_2, ω_1 and ω_2 satisfying that $\mathcal{BL}(x, t_1, \omega_1) = \mathcal{BL}(x, t_2, \omega_2)$, where $(t_1, \omega_1) \neq (t_2, \omega_2)$.

RP Designation w/ PID_{RP} Checking: There is no PID_{RP} collision. Given known $\mathbb{ID}_{RP}$, adversaries cannot find j_1, j_2, t_1, t_2, ω_1 and ω_2 which satisfy that $\mathcal{F}_{PID_{RP}}(ID_{RP_{j_1}}, t_1, \omega_1) = \mathcal{F}_{PID_{RP}}(ID_{RP_{j_2}}, t_2, \omega_2)$ where $j_1 \neq j_2$ and $1 \leq j_1, j_2 \leq p$.

The property is proposed for OPRF protocols: Given known $\mathbb{x}$, malicious OPRF users cannot find j_1, j_2, t_1, t_2, ω_1 and ω_2 satisfying that $\mathcal{BL}(x_{j_1}, t_1, \omega_1) = \mathcal{BL}(x_{j_2}, t_2, \omega_2)$ where $j_1 \neq j_2$ and $1 \leq j_1, j_2 \leq p$.

4.3.3 Privacy properties. User privacy of SSO services requires two properties as below, against an honest-but-curious IdP and malicious RPs, respectively.

IdP Untraceability: The *honest-but-curious* IdP learns nothing on the visited RP (i.e., ID_{RP}) from the token request (i.e., PID_{RP}).

IdP untraceability of a privacy-preserving SSO system built in Section 4.2, means that, the *honest-but-curious* server of the underlying OPRF learns nothing about x from x'.

RP Unlinkability: *Malicious* RPs cannot link any login initiated by an honest user visiting a malicious RP, to any subset of logins visiting any other colluding RPs by honest users, even when colluding with malicious users.

In an SSO system built based on an OPRF protocol, RP unlinkability is equivalent to *indistinguishability of keys* of the OPRF: When an OPRF protocol is initiated with different inputs, a malicious user cannot tell whether the OPRF server uses different keys or not across these protocol instances based on her collections, even if (*a*)

the OPRF user has collected some sets of protocol instances and (b) the instances in each collection set are generated by a certain but unknown key. Each protocol instance is composed of x, x' and the arguments and output of $\mathcal{UBL}()$, i.e., (x, x', z', t, z), along with ω if ω is an argument of $\mathcal{UBL}()$.[3] Every set of protocol instances collected by the malicious OPRF user, can be mapped to a user colluding with malicious RPs in SSO systems.

Firstly, indistinguishability of keys implies pseudo-randomness; otherwise, users could learn something on a key from z' and z, which might be exploited to distinguish different keys. On the other hand, pseudo-randomness does not ensure indistinguishability of keys, because existing OPRF protocols do not explicitly consider indistinguishable multiple keys [10]. For example, N is publicly known in 2HashRSA and usually uniquely identifies a key, but it is an argument of $\mathcal{UBL}(z', t) = H_2(x, z'/t \bmod N)$; or, even when the server generates one HE key pair for multiple OPRF keys in $\mathrm{DY_{HE}}$, $\omega = Enc(k)$ uniquely identifies a key unless a probabilistic HE scheme is adopted.

Key-Identifier Freeness of OPRFs is proposed accordingly: A protocol instance, i.e., x, x', and the arguments and output of $\mathcal{UBL}()$, cannot be exploited to distinguish an OPRF key from other keys. Thus, indistinguishability of keys is equivalent to both pseudo-randomness and key-identifier freeness of an OPRF tolerating malicious users.

These extended properties of OPRFs, *key-identifier freeness*, *RP designation* and *user identification*, have not been investigated in the literature [10], and do not always hold in all existing OPRFs.

4.4 Proofs of SSO-Related Properties in OPRFs

First of all, *IdP untraceability* against an *honest-but-curious* IdP in SSO is equivalent to the *obliviousness* property of an OPRFs working with an *honest-but-curious* OPRF server. All OPRF protocols satisfy this requirement.

RP unlinkability against malicious adversaries in SSO, is equivalent to *indistinguishability of keys* (i.e., pseudo-randomness and key-identifier freeness) of an OPRF tolerating malicious users. This property is analyzed as below and some protocols are accordingly revised.

- It holds in HashDH and $\mathrm{NR_{HE}}$ where pseudo-randomness is ensured with malicious users [24, 39], if there is an *identical* finite field $\mathbb{F}_q$ for all OPRF keys.
- Multiple keys are indistinguishable in $\mathrm{DY_{HE}}$, if (a) the server adopts an *identical* finite field $\mathbb{F}_q$ and (b) the adopted additively HE scheme is *probabilistic* or an *ephemeral* HE key pair is generated in each protocol instance.[4]
- Although pseudo-randomness is ensured with malicious users [39], key-identifier freeness does not hold in 2HashRSA because N uniquely identifies an OPRF key and is an argument of $\mathcal{UBL}()$. We slightly revise 2HashRSA as below, denoted as $\mathrm{2HashRSA}_N$ in this paper: An OPRF server generates multiple key pairs for an identical N, i.e., $e_i k_i = 1 \bmod \phi(N)$ for every key.

[3]These data are received by an RP in SSO with identity transformations [29, 30].

[4]Fortunately, the widely-used additively HE scheme, Paillier [53], is probabilistic. If a deterministic HE scheme is adopted and the HE key pair is permanent, an RP restores $\omega = Enc(k)$ by calculating $x'^{1/t}/Enc(x)$ and then it identifies an OPRF key.

Next, we analyze the properties for SSO security (i.e., user identification and RP designation) of these revised protocols.

Theorem 1. *User identification w/o PID_{RP} checking, is not ensured in $\mathrm{NR_{HE}}$ or $\mathrm{2HashRSA}_N$, but ensured in HashDH and $\mathrm{DY_{HE}}$.*

Proof. This property does not hold in $\mathrm{NR_{HE}}$. Because $|\mathbb{k}| = q^{n+1}$ but $|\mathbb{z}| = q$, $\hat{z}$ is equal to $\check{z}$ sometimes when $\hat{k} \neq \check{k}$. So user identification does not hold in $\mathrm{NR_{HE}}$, either without or with PID_{RP} checking.[5]

In $\mathrm{2HashRSA}_N$, $(e, k) \xleftarrow{\$} RSA(N)$, $x \xleftarrow{\$} \mathbb{Z}_N$, and $z = H_2(x, x^k)$. For any x, if $\hat{k} \neq \check{k}$, the probability that $H_2(x, x^{\hat{k}}) = H_2(x, x^{\check{k}})$ is negligible, due to collision-freeness of $H_2()$ and security of RSA. It also requires that OPRF users cannot find $\check{t}$ satisfying that $H_2(x, \hat{z}'/\check{t}) = H_2(x, x^{\check{k}})$, where $\check{k} \neq \hat{k}$. This property does not hold, because $\check{t} = \hat{t}x^{\hat{k}}/x^{\check{k}}$ satisfies that $H_2(x, \hat{z}'/\check{t}) = H_2(x, x^{\check{k}})$.

In HashDH, $k \xleftarrow{\$} \mathbb{F}_q$, $x \xleftarrow{\$} \mathbb{F}_q$, and $z = x^k$. Because x is a generator of $\mathbb{F}_q$, z is a bijective function of k. It further requires that, for any x, adversaries cannot find $\check{t}$ satisfying that $\hat{z}'^{1/\check{t}} = x^{\check{k}}$, i.e., $x^{\hat{t}\hat{k}/\check{t}} = x^{\check{k}}$, where $\check{k} \neq \hat{k}$. When $\mathbb{k}$ is unknown, this problem is equivalent to the discrete logarithm problem (DLP).

In $\mathrm{DY_{HE}}$, $k \xleftarrow{\$} \mathbb{F}_q$, $x \xleftarrow{\$} \mathbb{F}_q$, and $z = g^{1/(k+x)}$. Given x, $1/(k+x)$ is a bijective function of k. Because g is a generator of $\mathbb{F}_q$, $g^{1/(k+x)}$ is also a bijective function of k. It further requires that, for any x, OPRF users cannot find $\check{t}$ satisfying that $g^{\check{t}/\hat{t}(\hat{k}+x)} = g^{1/(\check{k}+x)}$, where $\check{k} \neq \hat{k}$. This problem is equivalent to the DLP. □

Next, we prove RP designation w/o PID_{RP} checking but do not consider $\mathrm{NR_{HE}}$ and $\mathrm{2HashRSA}_N$, for it is nonsense to discuss RP designation without user identification.

Theorem 2. *RP designation w/o PID_{RP} checking, is ensured in the OPRFs of HashDH and $\mathrm{DY_{HE}}$.*

Proof. In HashDH, given known $\mathbb{x}$, known $\mathbb{z}$, and unknown $\mathbb{k}$, if malicious OPRF users could find j_1, j_2, i_1, i_2, t_1 and t_2 satisfying that $((x_{j_1}^{t_1})^{k_{i_1}})^{1/t_2} = x_{j_2}^{k_{i_2}}$ where $j_1 \neq j_2$, the DLP would be solved. Thus, this property is ensured in HashDH.[6]

In $\mathrm{DY_{HE}}$ malicious OPRF users attempt to find t_1 and t_2 satisfying that $(g^{1/t_1(k_{i_1}+x_{j_1})})^{t_2} = g^{1/(k_{i_2}+x_{j_2})}$, i.e., $g^{t_1/t_2} = g^{(k_2+x_2)/(k_1+x_1)}$, where $j_1 \neq j_2$. It is equivalent to the DLP, when $\mathbb{k}$ is unknown. □

As user identification and RP designation w/o PID_{RP} checking are ensured in the OPRFs of HashDH and $\mathrm{DY_{HE}}$, weaker properties w/ PID_{RP} checking are certainly ensured in HashDH and $\mathrm{DY_{HE}}$. We omit these proofs in this paper.

Theorem 3. *User identification w/ PID_{RP} checking, holds in $\mathrm{2HashRSA}_N$.*

Proof. In $\mathrm{2HashRSA}_N$, for any x, if $\hat{k} \neq \check{k}$, the probability that $H_2(x, x^{\hat{k}}) = H_2(x, x^{\check{k}})$ is negligible due to collision-freeness of $H_2()$ and security of RSA. For any x, malicious OPRF users attempt to

[5]The NR OPRF using obviously transfer (OT) [24, 34] does not strictly conform to the formalization in Section 4.1, but this conclusion is also applicable to it.

[6]As mentioned in Section 2.1, both k_i and r_j are known to *only* the honest IdP, where $ID_{U_i} = k_i$ and $ID_{RP_j} = [r_j]G$. In the $\mathbb{F}_q$ versions of the identity transformations in UppreSSO, k_i and r_j are kept unknown to adversaries where $x_j = g^{r_j}$; otherwise, once $\mathbb{k}$ or $\mathbb{r} = \{r_{j=1,\cdots,p}\}$ is leaked, it is easy for adversaries to solve $(x_{j_1}^{t_1})^{k_{i_1}} = (x_{j_2}^{k_{i_2}})^{t_2}$.

Table 4: Properties of Typical OPRFs

OPRF Property	SSO-Related Property		HashDH	NR_{HE}	DY_{HE}	2HashRSA_N
Correctness	Correctness of Derived Accounts		√	√	√	√
Obliviousness	Privacy	IdP Untraceability	√	√	√	√
Indistinguishability of Keys[1]		RP Unlinkability	√	√	√	√
	Security w/o PID_{RP} Checking	User Identification	√	⊥	√	⊥
		RP Designation[2]	√		√	
	Security w PID_{RP} Checking	User Identification	√	⊥	√	√
		RP Designation[2]	√		√	√

√ means a property is ensured, and ⊥ means not.

1. Indistinguishability of keys = Pseudo-randomness + Key-identifier freeness. Some protocols are slightly revised in Section 4.4, to satisfy key-identifier freeness.
2. It is nonsense to discuss RP designation without user identification.

find t_1, t_2, e_{i_1} and e_{i_2} satisfying that $xt_1^{e_{i_1}} = xt_2^{e_{i_2}}$, i.e., $t_1 = (t_2^{e_{i_2}})^{d_{i_1}}$, where $(t_1, e_{i_1}) \neq (t_2, e_{i_2})$. It is equivalent to the decryption of given messages without the RSA private key. □

Theorem 4. *RP designation w/ PID_{RP} checking, is ensured in 2HashRSA_N.*

Proof. Given known $\mathbb{x}$, unknown $\mathbb{k}$, and known $\mathbb{e} = \{e_{i=1,\cdots,s}\}$, malicious OPRF users attempt to find j_1, j_2, i_1, i_2, t_1 and t_2 which satisfy that $x_{j_1} t_1^{e_{i_1}} = x_{j_2} t_2^{e_{i_2}}$, i.e., $t_1 (x_{j_1}/x_{j_2})^{k_{i_1}} = (t_2^{e_{i_2}})^{k_{i_1}}$, where $j_1 \neq j_2$. This problem cannot be solved due to security of RSA. □

4.5 Summary

Table 4 lists the properties of OPRF protocols. In addition to HashDH, DY_{HE} and 2HashRSA_N are qualified to work as the identity transformations in OIDC-compatible privacy-preserving SSO. That is, the identity-transformation approach proposed in UppreSSO can also be instantiated based on DY_{HE} and 2HashRSA_N, one of which does not require an RP to check PID_{RP} before accepting an identity token and the other does.

Finally, in the system utilizing 2HashRSA_N, when the visited RP checks whether PID_{RP} enclosed in a received token is equal to $x_j t^{e_i}$ or not, e_i leaks the user's identity and then RP unlinkability is broken. So only IdP untraceability is guaranteed in such a system, as BrowserID [22] and Poidc [31] do. Meanwhile, both RP unlinkability and IdP untraceability are guaranteed in the UppreSSO services built based on DY_{HE}. Note that some OPRF protocols, especially DY_{HE}, are slightly revised in Section 4.4, to satisfy key-identifier freeness.

We do not explicitly define this requirement, i.e., the calculation of $x' = \mathcal{BL}(x, t, \omega)$ cannot be exploited to distinguish an OPRF key from others, as an extended property of OPRFs. It is necessary for the visited RP to check whether PID_{RP} is equal to $\mathcal{BL}(x, t, \omega)$, *only* in the case that user identification and RP designation w/ PID_{RP} checking are ensured but these security-related properties w/o PID_{RP} checking do not hold.

5 Related Work

Security and Privacy of SSO Protocols. Sufficient conditions of secure SSO services are presented [21–23]: (*a*) An attacker cannot log into an honest RP as an account owned by any honest users, and (*b*) an honest user never log into an honest RP as an account not owned by this user. When authenticity, confidentiality and integrity of identity tokens are ensured, these conditions are equivalent to RP designation and user identification. Security of the SSO services with identity transformations has been proved as RP designation and user identification w/o PID_{RP} checking [30] or as the properties w/ PID_{RP} checking [29], but these variations are compared and analyzed only in this paper.

Dolev-Yao style models [21–23] are developed to analyze the communications among entities in an SSO system, to ensure that all messages including identity tokens are delivered as expected in the system and then to prove security and privacy of services based on the properties of traditional public-key and symmetric cryptographic algorithms (e.g., RSA and AES). On the contrary, security and privacy of SSO services are analyzed based on the complicated properties of different cryptographic primitives (i.e., OPRFs), while secure communications among entities (especially, authenticity, confidentiality and integrity of identity tokens) are assumed in this paper.

Indistinguishability is defined to analyze user privacy in Spresso [23], while privacy of SSO services integrating identity transformations is actually guaranteed by indistinguishability: All private inputs are indistinguishable to the OPRF server, and different OPRF keys are indistinguishable to users.

The identity transformations assume an honest IdP, while user privacy in SSO systems with a malicious IdP is considered in Spresso [23] and ticket transparency [12, 47]. This paper analyzes the relationship between the identity-transformation approach in SSO and OPRFs, and [4] discusses the mapping of security enhancements between SSO and X.509 certificate services.

OPRFs and OPRF-Based Applications. OPRFs [24, 34, 39, 42, 51] are designed and applied for password verification [25, 41], server-assisted encryption [6, 40], key recovery [3], computation on private inputs [17, 24, 34, 36], and anonymous tokens [14, 60]. S. Casacuberta, J. Hesse, and A. Lehmann systematized the knowledge of OPRF protocols [10].

Extended properties of OPRFs are proposed in various applications, including verifiability [13, 19, 39], committed inputs/outputs [8, 42], partial obliviousness [13, 19, 25], updateability [19], convertability [45] and extendability [13], but the OPRF properties related to privacy-preserving SSO (i.e., key-identifier freeness, RP

designation and user identification) are explicitly analyzed in this paper for the first time.

Privacy-Preserving Identity Federation. Identity federation [15, 37, 38, 49, 54, 62] enables a user registered at an IdP to be accepted by RPs, with different accounts, but it requires a user to maintain an extra long-term secret protecting accounts across RPs. If such an identity federation system is accessed from a browser, plug-ins or extensions need to be installed to process this secret. Although the same term "single sign-on (SSO)" was used [15, 49, 62], identity federation are different from OIDC-compatible SSO where a COTS browser acts as the user agent.

The solutions of identity federation prevent both IdP-based login tracing and RP-based identity linkage [15, 37, 38, 49, 54, 62], as (*a*) an IdP-issued anonymous credential does not encloses an RP's identity and (*b*) different pseudonyms are selected by a user to visit different RPs. They even protect user privacy against collusive attacks by the IdP and RPs, because the pseudonyms cannot be linked even if the ownership of anonymous credentials [7, 9, 11] is proved to RPs colluding with the IdP.

6 Conclusions and Future Work

In this paper, we investigate the identity-transformation approach of OIDC-compatible privacy-preserving SSO in two aspects: (*a*) The integration of identity transformations in an SSO system, with several suggestions to improve performance, and (*b*) the relationship between identity transformations in SSO and OPRFs, helping us to construct new qualified identity transformations for privacy-preserving SSO services constructed on top of OPRF protocols.

To the best of our knowledge, this is the first time to uncover the relationship between identity transformations in OIDC-compatible privacy-preserving SSO services and OPRFs, and prove the corresponding properties (i.e., key-identifier freeness, RP designation and user identification) of OPRFs, in addition to the basic properties of correctness, obliviousness and pseudo-randomness. These results greatly extend the understanding of both privacy-preserving SSO protocols and OPRFs.

In the future, we plan to integrate more efficient mechanisms (*a*) for a visited RP to be anonymously authenticated in the authorization code flow of OIDC and (*b*) for a user to obtain ID_{RP} of the visited RP. Meanwhile, we will study or design more OPRF protocols, and analyze their SSO-related properties to build privacy-preserving SSO services.

References

[1] M. Abdalla, M. Cornejo, A. Nitulescu, and D. Pointcheval. 2016. Robust password-protected secret sharing. In *21st European Symposium on Research in Computer Security (ESORICS)*.

[2] M. Asghar, M. Backes, and M. Simeonovski. 2018. PRIMA: Privacy-preserving identity and access management at Internet-scale. In *52nd IEEE International Conference on Communications (ICC)*.

[3] A. Bagherzandi, S. Jarecki, Y. Lu, and N. Saxena. 2011. Password-protected secret sharing. In *18th ACM Conference on Computer and Communications Security (CCS)*. 433–444.

[4] X. Bao, X. Zhang, J. Lin, D. Chu, Q. Wang, and F. Li. 2019. Towards the trust-enhancements of single sign-on services. In *3rd IEEE Conference on Dependable and Secure Computing (DSC)*.

[5] A. Bender, J. Katz, and R. Morselli. 2006. Ring signatures: Stronger definitions, and constructions without random oracles. In *3rd Theory of Cryptography Conference (TCC)*. 60–79.

[6] J. Camenisch, A. de Caro, E. Ghosh, and A. Sorniotti. 2019. Oblivious PRF on committed vector inputs and application to deduplication of encrypted data. In *23rd International Conference on Financial Cryptography and Data Security (FC)*.

[7] J. Camenisch and E. Herreweghen. 2002. Design and implementation of the Idemix anonymous credential system. In *9th ACM Conference on Computer and Communications Security (CCS)*.

[8] J. Camenisch and A. Lehmann. 2017. Privacy-preserving user-auditable pseudonym systems. In *2nd IEEE European Symposium on Security and Privacy (EuroS&P)*.

[9] J. Camenisch and A. Lysyanskaya. 2001. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*.

[10] S. Casacuberta, J. Hesse, and A. Lehmann. 2022. SoK: Oblivious pseudorandom functions. In *7th IEEE European Symposium on Security and Privacy (EuroS&P)*.

[11] M. Chase, S. Meiklejohn, and G. Zaverucha. 2014. Algebraic MACs and keyed-verification anonymous credentials. In *21st ACM Conference on Computer and Communications Security (CCS)*.

[12] D. Chu, J. Lin, F. Li, X. Zhang, and Q. Wang. 2019. Ticket transparency: Accountable single sign-on with privacy-preserving public logs. In *15th International Conference on Security and Privacy in Communication Networks (SecureComm)*.

[13] P. Das, J. Hesse, and A. Lehmann. 2022. DPaSE: Distributed password-authenticated symmetric-key encryption, or how to get many keys from one password. In *17th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*.

[14] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda. 2018. PrivacyPass: Bypassing Internet challenges anonymously. *Privacy Enhancing Technologies* 2018, 3 (2018), 164–180.

[15] A. Dey and S. Weis. 2010. PseudoID: Enhancing privacy for federated login. In *3rd Hot Topics in Privacy Enhancing Technologies (HotPETs)*.

[16] R. Dingledine, N. Mathewson, and P. Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*. 303–320.

[17] T. Duong and N. Trieu D.-H. Phan. 2020. Catalic: Delegated PSI cardinality with applications to contact tracing. In *ASIACRYPT*.

[18] J. Ernstberger, S. Chaliasos, G. Kadianakis, S. Steinhorst, P. Jovanovic, A. Gervais, B. Livshits, and M. Orru. 2024. zk-Bench: A toolset for comparative evaluation and performance benchmarking of SNARKs. In *14th International Conference on Security and Cryptography for Networks (SCN)*.

[19] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart. 2015. The Pythia PRF service. In *24th USENIX Security Symposium*.

[20] A. Faz-Hernandez, S. Scott, N. Sullivan, R. Wahby, and C. Wood. 2022. *draft-irtf-cfrg-hash-to-curve-16: Hashing to elliptic curves*. Internet Engineering Task Force.

[21] D. Fett, R. Küsters, and G. Schmitz. 2014. An Expressive Model for the Web Infrastructure: Definition and Application to the BrowserID SSO System. In *35th IEEE Symposium on Security and Privacy (S&P)*.

[22] D. Fett, R. Küsters, and G. Schmitz. 2015. Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the Web. In *20th European Symposium on Research in Computer Security (ESORICS)*.

[23] D. Fett, R. Küsters, and G. Schmitz. 2015. Spresso: A secure, privacy-respecting single sign-on system for the Web. In *22nd ACM Conference on Computer and Communications Security (CCS)*. 1358–1369.

[24] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. 2005. Keyword search and oblivious pseudorandom functions. In *2nd Theory of Cryptography Conference (TCC)*. 303–324.

[25] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. 2020. PESTO: Proactively secure distributed single sign-on, or how to trust a hacked server. In *5th IEEE European Symposium on Security and Privacy (EuroS&P)*.

[26] Google for Developers. [n. d.]. Google Identity: Integration considerations. https://developers.google.com/identity/gsi/web/guides/integrate/. Accessed January 13, 2025.

[27] P. Grassi, E. Nadeau, J. Richer, S. Squire, J. Fenton, N. Lefkovitz, J. Danker, Y.-Y. Choong, K. Greene, and M. Theofanos. 2017. *SP 800-63C: Digital identity guidelines: Federation and assertions*. National Institute of Standards and Technology.

[28] C. Guo, F. Lang, Q. Wang, and J. Lin. 2021. UP-SSO: Enhancing the user privacy of SSO by integrating PPID and SGX. In *International Conference on Advanced Computing and Endogenous Security (ICACES)*.

[29] C. Guo, J. Lin, Q. Cai, W. Wang, F. Li, Q. Wang, J. Jing, and B. Zhao. 2022. UppreSSO: Untraceable and unlinkable privacy-preserving single sign-on services (version 2). https://arxiv.org/abs/2110.10396.

[30] C. Guo, J. Lin, Q. Cai, W. Zhu, W. Wang, J. Jing, Q. Wang, B. Zhao, and F. Li. 2025. UppreSSO: Untraceable and unlinkable privacy-preserving single sign-on services (version 3). https://arxiv.org/abs/2110.10396.

[31] S. Hammann, R. Sasse, and D. Basin. 2020. Privacy-preserving OpenID Connect. In *15th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. 277–289.

[32] T. Hardjono and S. Cantor. 2018. *SAML v2.0 subject identifier attributes profile version 1.0*. OASIS.

[33] D. Hardt. 2012. *RFC 6749: The OAuth 2.0 authorization framework*. Internet Engineering Task Force.

[34] C. Hazay and Y. Lindell. 2008. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *5th Theory of Cryptography Conference (TCC)*.

[35] J. He, L. Lei, Y. Wang, P. Wang, and J. Jing. 2024. ArpSSO: An OIDC-compatible privacy-preserving SSO scheme based on RP anonymization. In *29th European Symposium on Research in Computer Security (ESORICS)*.

[36] A. Heinrich, M. Hollick, T. Schneider, M. Stute, and C. Weinert. 2021. PrivateDrop: Practical privacy-preserving authentication for Apple AirDrop. In *USENIX Security Symposium*.

[37] Hyperledger Fabric. [n. d.]. MSP implementation with Identity Mixer. https://hyperledger-fabric.readthedocs.io/en/release-2.2/idemix.html. Accessed July 20, 2022.

[38] M. Isaakidis, H. Halpin, and G. Danezis. 2016. UnlimitID: Privacy-preserving federated identity management using algebraic MACs. In *15th ACM Workshop on Privacy in the Electronic Society (WPES)*. 139–142.

[39] S. Jarecki, A. Kiayias, and H. Krawczyk. 2014. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *AsiaCrypt*.

[40] S. Jarecki, H. Krawczyk, and J. Resch. 2019. Updatable oblivious key management for storage systems. In *26th ACM Conference on Computer and Communications Security (CCS)*. 379–393.

[41] S. Jarecki, H. Krawczyk, and J. Xu. 2018. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In *EUROCRYPT*.

[42] S. Jarecki and X. Liu. 2009. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *6th Theory of Cryptography Conference (TCC)*. 577–594.

[43] M. Kroschewski and A. Lehmann. 2023. Save the implicit flow? Enabling privacy-preserving RP authentication in OpenID Connect. *Privacy Enhancing Technologies* 2023, 4 (2023), 96–116.

[44] M. Lee and S. Cho. 2013. *Web certificate API.* World Wide Web Consortium (W3C).

[45] A. Lehmann. 2019. ScrambleDB: Oblivious (Chameleon) Pseudonymization-as-a-Service. *Privacy Enhancing Technologies* 2019, 3 (2019), 289–309.

[46] W. Li and C. Mitchell. 2016. Analysing the security of Google's implementation of OpenID Connect. In *13th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*.

[47] G. Liu, J. Lin, D. Chu, X. Zhang, Q. Wang, C. Ma, F. Li, and D. Ye. 2023. Enhanced ticket transparency (eTT) framework for single sign-on services with pseudonyms. In *22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*.

[48] W. Ma, Q. Xiong, X. Shi, X. Ma, H. Jin, H. Kuang, M. Gao, Y. Zhang, H. Shen, and W. Hu. 2023. GZKP: A GPU accelerated zero-knowledge proof system. In *28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 340–353.

[49] G. Maganis, E. Shi, H. Chen, and D. Song. 2012. Opaak: Using mobile phones to limit anonymous identities online. In *10th International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

[50] E. Maler and D. Reed. 2008. The venn of identity: Options and issues in federated identity management. *IEEE Security & Privacy* 6, 2 (2008), 16–23.

[51] P. Miao, S. Patel, M. Raykova, K. Seth, and M. Yung. 2020. Two-sided malicious security for private intersection-sum with cardinality. In *CRYPTO*.

[52] M. Naor and O. Reingold. 2004. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM* 51, 2 (2004), 231–262.

[53] P. Paillier. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*.

[54] C. Paquin. 2013. *U-Prove technology overview v1.1.* Microsoft Corporation.

[55] D. Pointcheval and O. Sanders. 2016. Short randomizable signatures. In *The Cryptographers' Track at the RSA Conference (CT-RSA)*.

[56] N. Sakimura, J. Bradley, and N. Agarwal. 2015. *RFC 7636: Proof key for code exchange by OAuth public clients.* Internet Engineering Task Force.

[57] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore. 2014. *OpenID Connect core 1.0 incorporating errata set 1.* The OpenID Foundation.

[58] M. Thomson and C. Wood. 2024. *RFC 9458: Oblivious HTTP.* Internet Engineering Task Force.

[59] Uber Developers. [n. d.]. OIDC Web SDK. https://developer.uber.com/docs/consumer-identity/oidc/web. Accessed April 10, 2025.

[60] Web Incubator CG. [n. d.]. TrustToken API. https://github.com/WICG/trust-token-api. Accessed July 20, 2022.

[61] R. Xu, S. Yang, F. Zhang, and Z. Fang. 2023. Miso: Legacy-compatible privacy-preserving single sign-on using trusted execution environments. In *8th IEEE European Symposium on Security and Privacy (EuroS&P)*.

[62] Z. Zhang, M. Król, A. Sonnino, L. Zhang, and E. Rivière. 2021. EL PASSO: Efficient and lightweight privacy-preserving single sign on. *Privacy Enhancing Technologies* 2021, 2 (2021), 70–87.