OPPID: Single Sign-On with Oblivious Pairwise Pseudonyms

Maximilian Kroschewski
Hasso Plattner Institute,
University of Potsdam
maximilian.kroschewski@hpi.de

Anja Lehmann
Hasso Plattner Institute,
University of Potsdam
anja.lehmann@hpi.de

Cavit Özbay
Hasso Plattner Institute,
University of Potsdam
cavit.oezbay@hpi.de

ABSTRACT

Single Sign-On (SSO) allows users to conveniently authenticate to many Relying Parties (RPs) through a central Identity Provider (IdP). SSO supports unlinkable authentication towards the RPs via *pairwise pseudonyms*, where the IdP assigns the user an RP-specific pseudonym. This feature has been rolled out prominently within Apple's SSO service. While establishing unlinkable identities provides privacy towards RPs, it actually emphasizes the main privacy problem of SSO: with every authentication request, the IdP learns the RP that the user wants to access. Solutions to overcome this limitation exist, but either assume users to behave honestly or require them to manage long-term cryptographic keys.

In this work, we propose the first SSO system that can provide such pseudonymous authentication in an *unobservable* yet strongly secure and convenient manner. That is, the IdP blindly derives the user's pairwise pseudonym for the targeted RP without learning the RP's identity and without requiring key material handled by the user. We formally define the desired security and privacy properties for such unlinkable, unobservable, and strongly secure SSO. In particular, our model includes the often neglected RP authentication: the IdP typically wants to limit its services to registered RPs only and thus must be able to (blindly) verify that it issues the token and pseudonym to such a registered RP. We propose a simple construction that combines signatures with efficient proofs-of-knowledge with a blind, yet verifiable, evaluation of the Hashed-Diffie-Hellman PRF. We prove the security of our construction and demonstrate its efficiency through a prototypical implementation, which requires a running time of 2-12ms per involved party.

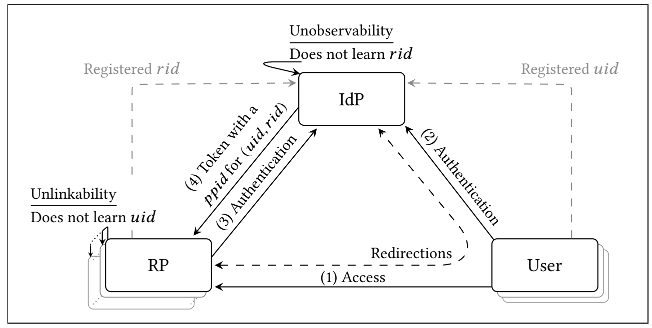

Figure 1: OPPID**: Users authenticate to RPs through the IdP. The IdP cannot observe which RP the user accesses. Users are unlinkable via RP-specific pseudonyms** ***ppids*****.**

1 INTRODUCTION

Single Sign-On (SSO) allows users to conveniently authenticate towards multiple online services with the help of a central party, the Identity Provider (IdP). When accessing a service – denoted as a Relying Party (RP) – users are redirected for authentication to the IdP. The IdP then verifies the user and sends a cryptographically signed token attesting the user's identity uid to the RP. The SSO approach frees users from the burden of remembering dedicated login credentials for each service they want to use, while also providing stronger authentication and simpler deployment for the RPs. Due to these characteristics, SSO has been widely adopted in recent years, particularly with major platform providers such as Google, Meta, or Apple serving as IdPs [1, 2, 20].

Unlinkability via Pseudonyms. A privacy drawback of SSO systems is that users become linkable across RPs through their identity uid, included in each token the IdP signs. Therefore, NIST recommends the use of *Pairwise Pseudonymous Identifiers* [37, §6.2.5] – short $ppid$. The IdP then replaces the user's identity uid in the token with a unique pseudonym $ppid$, which is derived *specifically* for the targeted RP. This protocol feature is supported by the widely-adopted OpenID Connect (OIDC) standard [43, §8], which uses a hash function H to set $ppid = \mathsf{H}(k, uid, rid)$, where k is a high-entropy key of the IdP and uid, rid are the identifiers of the user and RP, respectively. As the IdP assigns deterministic and unique pseudonyms for each user-RP combination, the RP is still ensured that the correct user logs in, and the same user cannot authenticate under multiple pseudonyms, which is known as Sybil-resistance. At the same time, the user can engage with different RPs under unlinkable pseudonyms, which has been prominently advertised by Apple in their *Sign in with Privacy* service [4] that uses this feature.

Main Challenge: Unobservability. While unlinkable pseudonyms improve user privacy with RPs, they highlight another fundamental SSO privacy problem: the IdP must know the RP's identity rid at each login to generate the user's RP-specific pseudonym. The pseudonym computation is not the only reason rid is revealed to the IdP in every authentication request. The most important purpose is to bind the token to the targeted RP for phishing protection, which is done by simply including rid in the signed token. Further, the IdP typically limits its service to *registered* RPs only, which requires some form of authentication from the RP to the IdP too [43].

The lack of unobservability is a significant risk to users' privacy. The IdP is involved in every online authentication and learns exactly which services and websites users access and when. As SSO is convenient for RPs when only a few IdPs exist, as is currently the case with Google, Meta, and Apple dominating the end-user SSO market, this concentration of information is particularly dangerous.

Thus, an important question is how the convenience and security of SSO can be provided in an *unobservable* manner. This requires that the IdP does not learn the targeted rid with every request but can still bind the signed token to the properly authenticated RP and support unlinkability through RP-specific pseudonyms. To maintain convenience for end-users, this should be in the plain SSO setting, i.e., not relying on any long-term keys or cryptographic credentials managed by the user.

The requirement for unobservability in federated identity management systems has already been discussed by Pfitzmann and Waidner in 2002 [40]. Their analysis underscores not only the need for unobservability, but also for pseudonymous user authentication that prevents the disclosure of stable user identifiers to multiple RPs, providing unlinkability across services. They also emphasize that while the IdP must authenticate RPs before issuing tokens, the IdP should not learn the exact access patterns of users. Two decades later, these principles remain highly relevant, particularly in the context of today's SSO landscape where major platform providers serve as IdP and gain access to detailed user behavior data.

Partial Solutions Towards Unobservable SSO. Realizing the requirements of unobservability, unlinkable pseudonym support, and RP authentication in SSO has been addressed in surprisingly few works, and all provide only partial solutions to the problem.

The first work to provide unlinkability and unobservability for users in OIDC was done by Hammann, Sasse, and Basin [28]. Their protocol, denoted as Pairwise POIDC (PPOIDC) [28], lets the IdP *blindly* bind the token to the targeted RP by signing a cryptographic commitment to rid. The pseudonym computation (via hash functions) is mostly outsourced to the user and again lets the IdP only blindly sign the pseudonym through a commitment. While the protocol ensures the correctness of the uid in the committed pseudonym by requiring the user to show it with a zero-knowledge proof, this is not the case for the rid. The protocol allows corrupt users to generate arbitrary IdP-attested pseudonyms per RP.

The UPPRESSO protocol by Guo et al. [27] also aims at pseudonymous SSO and generates pseudonyms through blind exponentiation of an rid-specific group element, enabling the RP to verify that it received a correctly computed pseudonym on its rid. The protocol focuses solely on the pseudonym, though, and does not detail how the final token is also strictly bound to the rid.

Further, both protocols do not support RP authentication towards the IdP. They only realize a weaker form, where the verification of the RP's legitimacy is outsourced to the user. Apart from putting more burden on the user, this also implicitly assumes that users must behave honestly. If a user misbehaves, or the user-side verification is not handled properly, the IdP can be tricked into providing its service to malicious and non-registered RPs or sign tokens that assert pseudonymous identities that are incorrect.

The first work to address privacy-preserving RP authentication directly to the IdP was recently done by Kroschewski and Lehmann [32]. Their AIF-ZKP (Authenticated Implicit Flow) protocol ensures that the IdP-issued token is bound to the intended and *authenticated* RP without disclosing rid to the IdP. While this approach provides unobservability towards the IdP, the protocol did not provide support for pseudonyms, i.e., it lacks unlinkability.

Thus, there is no protocol – or even security model – for such a fully private yet strongly secure SSO system.

Concrete Use Case: European Digital Identity Wallet. Apart from general SSO, there is also a more concrete use case that explicitly demands user authentication with unlinkability, unobservability, and RP authentication: the European Digital Identity Wallet. This Identity Wallet is part of the EU's eIDAS regulation, which came into effect in May '24 [19], and aims to establish government-attested and verifiable digital identities with the following requirements:

> *"Enable privacy-preserving techniques which ensure* ***unlinkability*** *[. . .] [19, §16b] — possibility of users to access services through the use of* ***pseudonyms*** *[. . .] [19, §22] — providers should ensure* ***unobservability*** *by not collecting data and not having insight into the transactions of the users [. . .] [19, §32] — relying parties should provide the information necessary to allow for their identification and* ***authentication*** *[. . .] [19, §17]"*

Every EU member state is now tasked with developing such an Identity Wallet for all its citizens and residents, creating an urgent demand for suitable technical solutions.

1.1 Our Contributions

In this work, we introduce the first SSO system (OPPID) that combines all properties of unlinkable and unobservable user authentication via a central IdP towards an authenticated RP. More specifically, we propose a protocol where the IdP issues its users strictly RP-bound tokens for a properly authenticated RP and containing RP-specific pseudonyms, yet learns nothing about the RP's identity. Our protocol achieves its security and privacy properties in a very convenient way, as it still works in the plain SSO setting, i.e., not relying on additional user-managed key material.

Formal Security Model for OPPID. The first core challenge is to properly define the security and privacy properties of this 3-party protocol, where each party has complementing views as depicted in Fig.1. In fact, neither of the aforementioned works on pseudonymous SSO provided a formal security model. We formalize *Unlinkability* and *Unobservability* as the two privacy properties, and security is expressed through notions of *Session Binding* and *Request Authentication*. The latter three build upon the model of [32]. The new property of Unlinkability – demanding that two corrupt RPs receiving pseudonymous user authentication cannot decide whether they interact with the same user or not – must carefully exclude trivial wins exploiting the deterministic nature of pseudonyms and their blind computation. Security expressed through Session Binding must hold despite unobservability, in particular guaranteeing that the user can only authenticate under correct and unique pseudonyms $ppid = \mathsf{F}(uid, rid)$ towards an authenticated RP – but where the IdP must not learn anything about rid. Our *Session Binding* definition builds upon [32] and discovers and fixes a weakness in their model: to balance security and unobservability, they guarantee Session Binding for *honest* users only, as this allows knowing the RP they intend to authenticate. However, this excludes the most important corruption setting. Thus, beyond extending their Session Binding notion to pseudonymous and unlinkable authentication, we strengthen their model by capturing security for malicious users.

Provably Secure Protocol π_{OPPID}. We propose a protocol that securely realizes all required properties. Our solution builds upon the SSO protocol with privacy-preserving RP authentication from [32] and shows how oblivious – yet strictly binding – pseudonym computation can be added. In a nutshell, [32] uses anonymous credentials for the RP's authentication towards the IdP and lets the IdP sign a verified commitment on rid in its token. To extend this to blindly computed pseudonyms, we rely on a variant of the HashDH (O)PRF [30] to realize $\mathsf{F}(uid, rid)$. While it is currently not known how such an oblivious PRF can be evaluated on blinded yet *verified inputs* – which would allow ensuring that pseudonyms are computed for the correct *rid* – we circumvent this missing building block: letting the IdP bind non-verified and verified *rid*-derived values in the signed (blinded) token and carefully checking for their consistency in the final token verification, where the *rid* is no longer blind. Thus, we can carry the guarantees from the verified *rid*-bound values over to the ones the IdP had to sign fully blind and ensure that valid tokens contain properly authenticated pseudonyms $ppid = \mathsf{F}(uid, rid)$, even with malicious users and RPs.

Implementation and Evaluation. To demonstrate the efficiency of our solution, we implemented our protocol using PS signatures [41] and Pedersen commitments [39] for RP authentication, RSA signatures for IdP tokens, and HashDH-style pseudonym computation in the PS signature source group. Our scheme is significantly faster than PPOIDC [28], requiring only 2-12ms per party. We report on the benchmarks of our open-source implementation and compare it in more detail to the closest related works.

Benefits of OPPID. Our work overcomes the primary privacy limitation of standard SSO systems: the exposure of users' access patterns to the IdP, while preserving the security and usability benefits of SSO. In particular, OPPID is the first protocol to achieve unlinkable and unobservable authentication without requiring users to manage cryptographic keys or credentials. However, the convenience of OPPID comes with an inherent privacy limitation: *colluding* IdP and RPs can trace users. In Sec. 1.2 we therefore discuss solutions that additionally provide such untraceability, but at the cost of requiring users to manage cryptographic keys or credentials. Such user-centric solutions demand users to take care of secure storage, synchronization across devices, and backup/recovery mechanisms. These challenges pose significant barriers to the deployment of user-held credential systems [33] and therefore limit the adoption of privacy-preserving solutions that rely on such a setting. Our OPPID solution presents an alternative path to privacy, enabling users who prefer a more convenient setup than credential-based solutions to enjoy the optimal privacy guarantees in such a setting.

Apart from usability, our OPPID has another benefit: it integrates explicit and IdP-controlled RP authentication, an aspect that has received limited attention in user-centric systems, such as anonymous credentials, yet. Ensuring proper RP authentication is necessary to avoid over-identification, which can threaten users' privacy regardless of the deployed cryptography. Such RP authentication is notably harder to achieve in user-centric systems, where users bear the sole responsibility of verifying an RP's legitimacy.

The fact that OPPID does not require any key material handled by the user makes it also particularly suited for browser-based, zero-footprint scenarios, as introduced by Pfitzmann and Waidner [40]. Such scenarios demand that protocols operate across common web browsers on standard devices, without requiring any additional software or pre-configured components. By operating entirely within the plain SSO model, OPPID ensures compatibility and ease of use across devices without imposing additional technical requirements on users.

1.2 Related Work

We have already mentioned the related work that is closest to ours: PPOIDC [28], UPPRESSO [27], and AIF-ZKP [32], all of which operate within the plain SSO model. We consider the plain SSO model as one where users do not manage long-term keys or credentials, crucial for convenience and adoption, but this comes with privacy limitations: *colluding* IdP and RPs can trace users. Therefore, we briefly discuss solutions for pseudonymous user authentication that provide stronger privacy than our work, but at the cost of reduced usability. A summary of pseudonymous user authentication solutions and a comparison to our work is given in Tab. 1.

Protocols Outside the Plain SSO Model. The protocols [16, 23, 26, 45, 46] provide untraceable pseudonymous authentication but either introduce additional parties or rely on user-managed secret keys, thus deviating from the plain SSO setting PseudoID [16] introduces an additional token service to blindly sign a token that gets bound to a pseudonym and user secret, allowing users to authenticate directly to an RP. Besides the extra party, this approach makes RP authentication towards the IdP impossible due to the token's independence from the RP's identity. EL PASSO [46] lets users obtain a short-lived anonymous credential from the IdP, again bound to a user-held key. The user can then locally derive an RP-specific pseudonym and presentation token from that credential and key for each login. This provides untraceable authentication but again detaches the RP authentication from the IdP and requires users to manage a long-term key. PrivSSO [23] requires users to create and manage a dedicated signature key pair for each RP account, which is then bound to a generic IdP token. Using both enables pseudonymous and unobservable authentication towards an RP but relies on even more keys that need to be securely stored and orchestrated by the user. The approaches [26, 45] do not require user keys but leverage secure enclaves on the user side or on an extra party to compute the users' pseudonyms. The enclave acts as an intermediary between the RPs and the IdP, ensuring pseudonym correctness through remote attestation and eliminating the need for the IdP to learn the *rid*.

Anonymous Credentials. An alternative to SSO-based solutions are fully user-centric systems, such as anonymous credentials, e.g., [12, 13, 38, 44]. In these systems, users receive long-term credentials from a trusted IdP, based on their personal attributes, which they can use autonomously to authenticate with RPs. Since the IdP is not involved in individual authentication sessions, *unobservability* is guaranteed by design. Additionally, anonymous credentials support *unlinkable* presentations, enabling users to selectively disclose their information in a pseudonymous manner.

In terms of privacy, anonymous credentials are superior to any (plain) SSO-based system, as they additionally provide *untraceability* even when the IdP and RP collude. We discuss that this property is impossible to achieve in *any* plain SSO system that supports RP-specific pseudonyms in Sec. 3.4. However, the price for better privacy is the additional responsibility placed on users. This burden to securely manage cryptographic keys and credentials, limited the adoption of anonymous credentials.

Relation to Domain-Pseudonyms. The concept of pseudonyms that are unique and consistent per RP but unlinkable across different RPs also exist in anonymous credential and Direct Anonymous Attestation (DAA) systems [8, 10, 13, 35]. These pseudonyms are referred to as scope-exclusive or domain pseudonyms, and as basenames in DAA. Our concrete instantiation relies on the same Diffie-Hellman function $\mathsf{PRF}(k, x) = H(x)^k$ that has been used for pseudonyms in discrete logarithm-based credential constructions, such as [9, 17, 35]. As the Diffie-Hellman construction is known to be a secure pseudorandom function, it naturally lends itself to this purpose. In anonymous credential systems, pseudonyms are computed as $nym = H(rid)^{usk}$ by the user, who possesses the secret key usk and a credential on usk from the IdP. For each pseudonymous authentication, the credential is used to compute a zero-knowledge proof that the pseudonym is well-formed for the targeted rid, ensuring correctness and Sybil resistance. While our construction uses the same PRF, the computation process and correctness guarantees differ entirely: in our approach, the key usk is known only to the IdP, not the user. The IdP must compute the PRF value, but neither learn rid nor the pseudonym nym it derives, yet also guarantee correctness of its computation towards the RP. This is achieved using the oblivious nature of the Diffie-Hellman PRF and carefully leveraging the individual knowledge each party holds. For a more detailed discussion on the challenges of computing these pseudonyms and our solutions, we refer to Sec. 4.2.

2 SSO WITH OBLIVIOUS PPIDS

Before we present our pseudonymous SSO system OPPID, we introduce its entities and detail the properties of pseudonymous user authentication. Our system builds upon the standard SSO model, where this privacy mechanism is commonly realized via a Pairwise Pseudonymous Identifier, as outlined by NIST [37, §6.2.5] and further specified by OIDC [43, §8].

2.1 Entities & Main Phases

OPPID protocol is built in the classic SSO setting with three entities: Users, Relying Parties (RPs), and a central Identity Provider (IdP):

Users: The user is registered with the IdP under a unique username uid. We assume the IdP handles all user-related registration and authentication but omit those details from our model. For our purposes, the crucial part is that the user is known as uid to the IdP but has individual pseudonyms $ppid$ for each Relying Party.

RPs: The RP is the service the user wishes to access and known as rid to the user and IdP. The RP relies on the IdP for user authentication and for receiving additional user and session information ctx. The RP knows the user only under their pseudonym $ppid$. To use the IdP's service, the RP must be registered with the IdP.

IdP: The IdP is the central authority that RPs and users rely on for authentication. It issues a token τ, which asserts to an RP that it is communicating with the user known as $ppid$. Apart from the pseudonym, the token is also bound to a particular session referenced by sid, additional user/session data ctx, and the targeted RP rid. While we do not detail how users authenticate to the IdP, our model explicitly covers that only registered and authenticated RPs can use the IdP's service.

As one of our primary requirements is proper RP authentication, we roughly divide our system into two phases:

Phase 1: RP Registration. An RP must register with the IdP before using the IdP's authentication service. We assume that each RP has a unique identifier rid and denote with $\mathcal{M}$ the set of registered RPs.

Phase 2: Authentication. When users with a unique username uid want to authenticate with a specific RP rid, they start the authentication session with the targeted rid, but without revealing their username to the RP. The RP then provides authentication information $auth$ and a session identifier sid, which are sent to the IdP via the user. When forwarding $sid, auth$ to the IdP, the user now reveals uid to the IdP, and we assume that the IdP has the means to check whether the user uid is correctly authenticated.

We do not detail how the user authenticates to the IdP, but we require that the IdP checks that the request stems from a registered RP, i.e., $rid \in \mathcal{M}$. If so, the IdP generates a token τ that pseudonymously authenticates the user uid as $ppid = \mathsf{F}(uid, rid)$ towards rid, where F is a pseudonym function we detail next. The final token τ_{fin} must be strictly bound to $rid, ppid, sid$ and a context ctx, which denotes additional session/user information certified by the IdP.

2.2 Pairwise Pseudonymous Identifier

Our system focuses on providing the Pairwise Pseudonymous Identifier ($ppid$) feature of OIDC [43] that hides the user's uid from an RP. The core properties of the $ppid$ are:

Uniqueness: For every combination of rid and uid, there exists a unique mapping to a $ppid$. We model this by assuming the $ppid$ to be derived through a *deterministic* function F as $ppid = \mathsf{F}(uid, rid)$.

Collision Freeness: For every rid and for all $uid \neq uid'$, it must hold that $\mathsf{F}(uid, rid) \neq \mathsf{F}(uid', rid)$, i.e., different users are assigned different pseudonyms towards the same RP rid.

Unlinkable Pseudonyms: Seeing two pseudonyms for different $rid_0 \neq rid_1$ with $ppid_0 = \mathsf{F}(uid, rid_0)$ and $ppid_1 = \mathsf{F}(uid', rid_1)$, it is infeasible to determine whether $uid = uid'$ or not.

As F is deterministic and the set of user names typically small, the property of unlinkable pseudonyms requires that F must not be known to the RPs seeing the user's pseudonyms. This is typically realized by using a pseudorandom function PRF for F, where the key k is only known to the IdP. (While collision-freeness does not follow directly from the pseudorandomness of PRF, it is implied if the function is injective and has a larger range than domain).

Our work addresses the challenge of enabling (partially) blind – yet authenticated – $ppid$ computation. Specifically, the IdP knows uid but not rid, while ensuring it computes valid tokens for $\mathsf{F}(uid, rid)$ for the targeted and properly authenticated RP rid.

2.3 Syntax of OPPID

We present the syntax of OPPID, *Oblivious Pairwise Pseudonymous Identifier*, protocol and provide an overview of the authentication process in Fig. 2. More explanation on how OPPID is used in the SSO flow and the correctness definition can be found in App. A.

Definition 2.1 (Syntax of OPPID*).* In more detail, an OPPID scheme is defined as a tuple of algorithms $(\mathsf{Setup}, \mathsf{KGen}_{\mathsf{IdP}}, \langle \mathsf{Join}_{\mathsf{RP}}, \mathsf{Reg}_{\mathsf{IdP}} \rangle, \mathsf{AInit}_{\mathsf{U}}, \mathsf{AReq}_{\mathsf{RP}}, \mathsf{ARes}_{\mathsf{IdP}}, \mathsf{AFin}_{\mathsf{U}}, \mathsf{Vf}_{\mathsf{RP}})$:

Approach \ Property	Privacy		Security		Other
	Unobservability	*Unlinkability*	*Req. Auth.*	*Session Binding*	*Plain SSO Model*
OIDC With Pseudonyms*[43]	○	●	●	●	●
PseudoID [16]	●	●	○	○	○
PPOIDC*[28]	●	●	○	◐	●
UPPRESSO*[27] / BISON [29]	●	●	○	◐	●
EL PASSO [46]	●	●	○	○	○
AIF-ZKP*[32]	●	○	●	●	●
MISO [45]	●	●	●	●	○
PrivSSO [23]	●	●	○	○	○
Our Work: OPPID	●	●	●	●	●

*Detailed security comparison given in Sec. 5

Table 1: Overview of SSO protocols, supporting RP authentication and/or pseudonymous user authentication.

$\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$ For a security parameter, returns the public parameters pp, which serve as implicit input for all other algorithms.

$\mathsf{KGen}_{\mathsf{IdP}}(pp) \rightarrow ((isk, \mathcal{M}), ipk)$ Returns the membership state $\mathcal{M}$ and the IdP key pair, isk and ipk.

$\langle \mathsf{Join}_{\mathsf{RP}}(ipk, rid), \mathsf{Reg}_{\mathsf{IdP}}(isk, rid, \mathcal{M})\rangle \rightarrow \{(cred, \mathcal{M}'), \bot\}$ An interactive protocol between the RP and IdP. Successful execution results in the RP acquiring a credential $cred$, and the IdP yielding an updated member state $\mathcal{M}'$. In case of failure, it returns $\bot$.

$\mathsf{AInit}_{\mathsf{U}}(ipk, rid) \rightarrow (orid, crid)$ Run by the user to initialize a token request via an IdP with ipk for RP rid. It returns a committing value $crid$ and an opening $orid$.

$\mathsf{AReq}_{\mathsf{RP}}(ipk, rid, cred, crid, orid, sid) \rightarrow auth$ Run by an RP, taking rid, credential $cred$, user commitment $crid$ and opening $orid$, and a random session sid. It returns the RP authentication $auth$.

$\mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid, ctx, sid) \rightarrow \{\tau, \bot\}$ Run by the IdP using it's secret key isk, RP authentication data $auth$, user commitment $crid$, context ctx, and session identifier sid. If the verification of the request fails, it outputs $\bot$ and a token τ otherwise.

$\mathsf{AFin}_{\mathsf{U}}(ipk, rid, crid, orid, ctx, sid, \tau) \rightarrow \{(\tau_{\mathsf{fin}}, ppid), \bot\}$ Run by the user taking rid, user commitment $crid$ and opening $orid$, context ctx, and sid to finalize the token τ. Returns $\bot$ if inputs are invalid, and otherwise the finalized token τ_{fin} and pseudonym $ppid$.

$\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid, ppid, ctx, sid), \tau_{\mathsf{fin}}) \rightarrow 0/1$ Returns 1 if τ_{fin} is valid under ipk for $(rid, ppid, ctx, sid)$ and otherwise 0.

3 SECURITY MODEL OF OPPID

We now formally define the privacy and security properties of an OPPID system, building upon the model of [32]. While we reuse some of their properties (Request Authentication and Unobservability which were called RP Accountability and RP Hiding in [32]), we also require the additional Unlinkability property and extend their Session Binding model to cover the pseudonymous authentication we aim for. Interestingly, the original model for Session Binding was rather weak, which we strengthen with our work too. We start with a high-level intuition of the desired properties and then present our formal model as game-based security definitions.

Note that some requirements for the pseudonym function F are already specified (see Sec. 2.2): each user must have a single pseudonym per rid (*Uniqueness*), and distinct users obtain different pseudonyms for the same rid (*Collision Freeness*). These two guarantees imply that F is deterministic and injective, so we omit an explicit formalization for these straightforward properties.

In addition to these basic pseudonym properties, we require their computation to be done in a blind way by the IdP (*Unobservability*), as well as the unlinkability of the pseudonymous authentication (*Unlinkability*), which also includes the unlinkability of F.

Unlinkability: The user's identity uid should remain hidden towards RPs – they should only know users under their RP-specific pseudonym $ppid$. This implies that when the same user authenticates to two different RPs as $ppid_0$ and $ppid_1$, the two RPs cannot distinguish whether they are communicating with the same user or two different users.

Unobservability: The RP's identity rid should remain hidden towards an IdP during the authentication session. This property assumes that RPs and users are honest and ensures privacy towards a potentially corrupt IdP.

Despite the blind $ppid$ computation and privacy-preserving RP authentication, IdP-issued tokens must stay unforgeable and bound to the blindly verified rid and the correctly computed pseudonym.

Session Binding: It is infeasible to create a valid token τ_{fin} for a session identified through $(rid, ppid, ctx, sid)$ that was not properly authenticated or approved by the honest IdP. This notion ensures that the user (and RP) can only generate tokens for the unique and correct $ppid = \mathsf{F}(uid, rid)$ they have jointly authenticated, where RPs must be properly registered with the IdP.

Kroschewski and Lehmann define the property of *Request Authentication* [32], which complements their Session Binding notion. While Session Binding expresses the security of the final token, this additional property demands that every valid request (including intermediate protocol values) processed by the IdP must come from a properly registered RP. As this property is unaffected by the blind pseudonym computation focused on in this work, we restate this notion in our setting and refer to the detailed explanation in App. A.

Oracles. Our definitions are given in a game-based notion, where an adversary $\mathcal{A}$ runs an experiment with a challenger who manages all honest entities and their private states. These interactions with

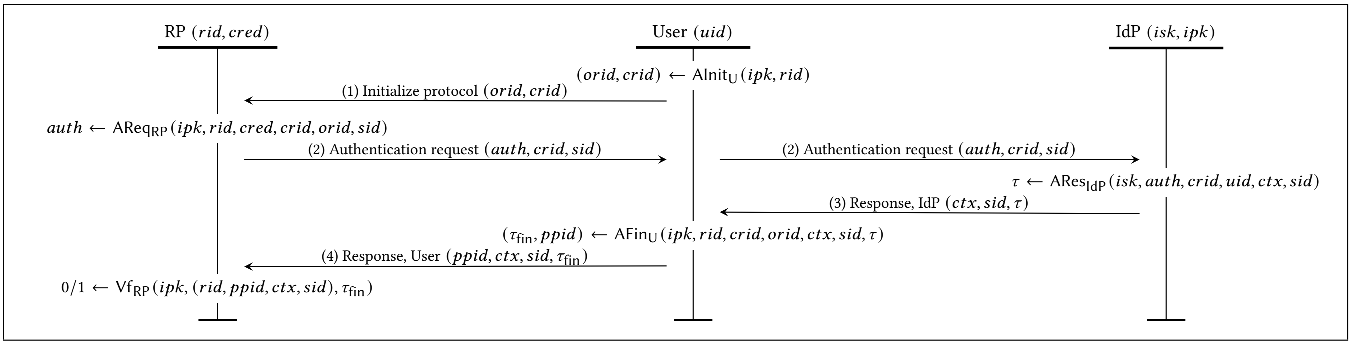

Figure 2: User authentication in OPPID **to a registered RP, which has previously obtained a credential** *cred* **from the IdP.**

honest entities are captured through oracles (see Fig. 3, right), which we briefly outline before presenting our security games.

We give the adversary the ability to register RPs with the IdP, where $\mathcal{A}$ runs the part of the corrupt party (either RP or IdP) and interacts with the honest counterpart through the oracle.

In our games with an honest IdP, RegHRP and RegCRP execute the registration with honest and corrupt RPs (run by $\mathcal{A}$), respectively. JoinCIdP takes the role of an honest RP that joins towards a corrupt IdP, which is only used in the Unobservability game.

We allow the adversary to intercept, capture, or inject messages between honest parties during an authentication session. $\mathsf{AInit}_\mathsf{U}$ initiates an honest user session which can later be finalized using the AResFin oracle. $\mathsf{AReq}_\mathsf{RP}$ returns an honest RP authentication *auth* for a session referenced by *sid*. AResFin lets $\mathcal{A}$ obtain a finalized token τ_{fin} and *ppid* from an honest user session initiated using the $\mathsf{AInit}_\mathsf{U}$ oracle. This oracle simulates secure communication between an honest user and honest IdP. $\mathsf{ARes}_\mathsf{IdP}$ allows $\mathcal{A}$ to request an IdP token on fully maliciously generated inputs. Vf_RP verifies a session $(rid, ppid, ctx, sid)$ against a finalized token τ_{fin} and the *ipk*. This is used to keep track of tokens shown by the adversary and detect "double-spending" (detailed in our Session Binding game).

3.1 Unlinkability

Unlinkability captures the core privacy feature of *pseudonymous* authentication, where authentication is done under a pseudonym *ppid* that is hiding the user's identity towards malicious RPs. This property requires the unlinkability of pseudonyms produced via F across RPs, which will be a convenient stepping stone in the formal analysis. Additionally, it ensures that authentication tokens do not leak any information about *uid* beyond the requested pseudonym.

We model this property through a classic indistinguishability experiment, which is defined through the game $\mathsf{Exp}^{\mathsf{UNLINK}}_{\mathcal{A},\mathsf{OPPID}}$ (see Fig. 3). In this game, the IdP is honest, and the adversary $\mathcal{A}$ controls all RPs. It can register RPs through $\mathcal{O}$.RegCRP and let honest users initiate sessions through $\mathcal{O}.\mathsf{AInit}_\mathsf{U}$, and receive tokens and RP-specific *ppid*s from the honest IdP via $\mathcal{O}.\mathsf{ARes}_\mathsf{IdP}$.

Eventually, $\mathcal{A}$ outputs two challenge users uid_0 and uid_1 along with common session information *sid*, *ctx*, *auth*, and *crid*. The game returns the token and $ppid_b$ for the randomly chosen uid_b, requiring the adversary to determine the bit b better than by guessing.

Excluding Trivial Wins. Since *ppid*s are deterministically (yet blindly) derived for every *uid*, *rid* combination, we must prevent trivial wins exploiting this determinism. Specifically, if the adversary has already learned $ppid_0$ or $ppid_1$ through interactions with the oracles, winning this game becomes trivial. Therefore, we ensure that $\mathcal{A}$ never learns these values through two abort conditions.

Before looking at these conditions, note that Unlinkability is meaningful and defined only for *honest* users. Thus, our challenger computes the pseudonym for an honestly generated *crid*, knowing the target RP *rid* for which the challenge pseudonym $ppid_b := \mathsf{F}(uid_b, rid)$ is computed.

The first check in our winning condition ensures that $(uid_d, rid) \notin \mathsf{Q}_{\mathsf{ppid}}$ for $d \in \{0, 1\}$, meaning the adversary never triggered either challenge user uid_d to initiate an honest session for *rid* (via $\mathcal{O}.\mathsf{AInit}_\mathsf{U}$ and $\mathcal{O}.\mathsf{ARes}_\mathsf{IdP}$), which would reveal $ppid_d$. Here, "honest" implies $crid_i$ was honestly generated in each request, allowing the challenger to precisely know *rid* and the $ppid_b$ that $\mathcal{A}$ learned.

When the adversary queries $\mathcal{O}.\mathsf{ARes}_\mathsf{IdP}$ with *crid* that was not honestly generated, the challenger lacks information about which *rid* the adversary requested the token and pseudonym for, requiring stricter abort conditions. This is captured by $\mathcal{O}.\mathsf{ARes}_\mathsf{IdP}$ keeping records (uid, adv) in $\mathsf{Q}_{\mathsf{ppid}}$ for such adversarial sessions, and later enforcing that $(uid_d, \mathsf{adv}) \notin \mathsf{Q}_{\mathsf{ppid}}$ for $d \in \{0, 1\}$. Here, adv denotes that the adversary cannot query $\mathcal{O}.\mathsf{ARes}_\mathsf{IdP}$ for *any* adversarial query concerning the challenge users. This is unavoidable as we cannot determine which $ppid = \mathsf{F}(uid, ?)$ $\mathcal{A}$ has obtained.

Capturing Unlinkability. Note that $\mathcal{A}$ can receive pseudonyms *ppid* for both challenge users uid_0 and uid_1 for all $rid' \neq rid$ apart from the target *rid* from the challenge query, while using $\mathcal{O}.\mathsf{AInit}_\mathsf{U}$. This capability is crucial to capture the desired *unlinkability* of the users' pseudonyms across RPs. Requiring $\mathcal{A}$ to use $\mathcal{O}.\mathsf{AInit}_\mathsf{U}$ merely mimics how *honest* users would behave, who we aim to protect with this property.

Definition 3.1 (Unlinkability). An OPPID scheme is unlinkable if for all PPT adversaries $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{UNLINK}}_{\mathcal{A},\mathsf{OPPID}}(\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

Content of ctx. We emphasize that in practice, privacy guaranteed by Unlinkability strongly depends on the information revealed in *ctx*. Our model assumes that *ctx* is identical for both uid_0 and uid_1, as revealing different *ctx* would render distinguishing the

Unlinkability : $\mathsf{Exp}^{\mathsf{UNLINK}}_{\mathcal{A},\mathsf{OPPID}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp)$; $b \leftarrow_{\mathsf{R}} \{0,1\}$
$O := \{\mathsf{RegCRP}, \mathsf{AInit}_{\mathsf{U}}, \mathsf{ARes}_{\mathsf{IdP}}\}$
$(uid_0, uid_1, auth, crid, ctx, sid) \leftarrow \mathcal{A}^{O}(ipk)$
Require $(rid, crid, orid) \in \mathsf{Q}_{\mathsf{rid}}$
For $d \in \{0,1\}$:
  $\tau_d \leftarrow \mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid_d, ctx, sid)$
  $(\tau_{fin_d}, ppid_d) \leftarrow \mathsf{AFin}_{\mathsf{U}}(ipk, rid, crid, orid, ctx, sid, \tau_d)$
  Require $\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid, ppid_d, ctx, sid), \tau_{fin_d}) = 1$
$b^* \leftarrow \mathcal{A}^{O}(\tau_{fin_b}, ppid_b)$
Abort if for $d \in \{0,1\}$: $(uid_d, rid) \in \mathsf{Q}_{\mathsf{ppid}} \lor (uid_d, \mathsf{adv}) \in \mathsf{Q}_{\mathsf{ppid}}$
Return 1 if $b = b^*$

Unobservability: $\mathsf{Exp}^{\mathsf{UNOBS}}_{\mathcal{A},\mathsf{OPPID}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp)$; $b \leftarrow_{\mathsf{R}} \{0,1\}$
$O := \{\mathsf{JoinCIdP}, \mathsf{AReq}_{\mathsf{RP}}\}$
$(rid_0, rid_1, sid) \leftarrow \mathcal{A}^{O}((isk, \mathcal{M}), ipk)$
For $d \in \{0,1\}$: Require $(rid_d, cred_d) \in \mathsf{HRP}$
$(orid, crid) \leftarrow \mathsf{AInit}_{\mathsf{U}}(ipk, rid_b)$
$auth \leftarrow \mathsf{AReq}_{\mathsf{RP}}(ipk, rid_b, cred_b, crid, orid, sid)$
$b^* \leftarrow \mathcal{A}^{O}(auth, crid)$
Return 1 if $b = b^*$

Request Authentication: $\mathsf{Exp}^{\mathsf{REQ\text{-}AUTH}}_{\mathcal{A},\mathsf{OPPID}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp)$
$O := \{\mathsf{RegHRP}, \mathsf{AReq}_{\mathsf{RP}}, \mathsf{ARes}_{\mathsf{IdP}}\}$
$(auth^*, crid^*, uid^*, ctx^*, sid^*) \leftarrow \mathcal{A}^{O}(ipk)$
Return 1 if $\mathsf{ARes}_{\mathsf{IdP}}(isk, auth^*, crid^*, uid^*, ctx^*, sid^*) \neq \bot \land$
 $(\cdot, crid^*, sid^*) \notin \mathsf{Q}_{\mathsf{auth}}$

Session Binding: $\mathsf{Exp}^{\mathsf{SES\text{-}BIN}}_{\mathcal{A},\mathsf{OPPID}}(\lambda)$

$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp)$
$O := \{\mathsf{RegHRP}, \mathsf{RegCRP}, \mathsf{AInit}_{\mathsf{U}}, \mathsf{AReq}_{\mathsf{RP}}, \mathsf{ARes}_{\mathsf{IdP}}, \mathsf{AResFin}, \mathsf{Vf}_{\mathsf{RP}}\}$
$(rid^*, ppid^*, ctx^*, sid^*, \tau^*_{\mathsf{fin}}) \leftarrow \mathcal{A}^{O}(ipk)$
Return 1 if $\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid^*, ppid^*, ctx^*, sid^*), \tau^*_{\mathsf{fin}}) = 1 \land$
(1) $(\cdot, ctx^*, sid^*) \notin \mathsf{Q}_\tau \lor$ // *Direct Forgery*
(2) $(uid, ctx^*, sid^*) \in \mathsf{Q}_\tau$ and at least one of the following holds:
 (a) $ppid^* \neq \mathsf{F}(uid, rid^*)$ // *Nym Correctness*
 (b) $(rid, uid, ctx^*, sid^*) \in \mathsf{Q}_{\tau_{\mathsf{fin}}} \land rid \neq rid^*$ // *RP Binding I*
 (c) $(rid, ppid, ctx^*, sid^*) \in \mathsf{Q}_{\mathsf{vf}} \land rid \neq rid^*$ // *RP Binding II*
 (d) $rid^* \notin \mathsf{HRP} \cup \mathsf{CRP}$ // *RP Authentication I*
 (e) $rid^* \in \mathsf{HRP} \land (rid^*, \cdot, sid^*) \notin \mathsf{Q}_{\mathsf{auth}}$ // *RP Authentication II*

Oracle : $\mathsf{RegHRP}(\mathsf{rid})$

Require $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$
// *Both RP and IdP are honest*
$\langle \mathsf{Join}_{\mathsf{RP}}(ipk, rid), \mathsf{Reg}_{\mathsf{IdP}}(isk, rid, \mathcal{M}) \rangle$
Upon output $(cred, \mathcal{M}')$
 $\mathsf{HRP} := \mathsf{HRP} \cup \{(rid, cred)\}$
Return 1

Oracle : $\mathsf{JoinCIdP}(\mathsf{ipk}, \mathsf{rid})$

Require $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$
// *$\mathcal{A}$ being the corrupt IdP*
Run $\mathsf{Join}_{\mathsf{RP}}(ipk, rid)$ with $\mathcal{A}$
Upon output $cred$
 $\mathsf{HRP} := \mathsf{HRP} \cup \{(rid, cred)\}$
Return 1

Oracle : $\mathsf{RegCRP}(\mathsf{rid})$

Require $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$
// *$\mathcal{A}$ being the corrupt RP*
$\mathsf{Reg}_{\mathsf{IdP}}(isk, rid, \mathcal{M})$ with $\mathcal{A}$
Upon output $\mathcal{M}'$
 $\mathsf{CRP} := \mathsf{CRP} \cup \{(rid, \cdot)\}$
Return 1

Oracle : $\mathsf{AInit}_{\mathsf{U}}(\mathsf{rid})$

$(orid, crid) \leftarrow \mathsf{AInit}_{\mathsf{U}}(ipk, rid)$
// *Req. for Unlinkability*
$\mathsf{Q}_{\mathsf{rid}} := \mathsf{Q}_{\mathsf{rid}} \cup \{(rid, crid, orid)\}$
Return $(orid, crid)$

Oracle : $\mathsf{AReq}_{\mathsf{RP}}(\mathsf{rid}, \mathsf{crid}, \mathsf{orid}, \mathsf{sid})$

Require $(rid, cred) \in \mathsf{HRP}$
$\mathsf{Q}_{\mathsf{auth}} := \mathsf{Q}_{\mathsf{auth}} \cup \{(rid, crid, sid)\}$ // *Req. for Session Binding*
Return $auth \leftarrow \mathsf{AReq}_{\mathsf{RP}}(ipk, rid, cred, crid, orid, sid)$

Oracle : $\mathsf{ARes}_{\mathsf{IdP}}(\mathsf{auth}, \mathsf{crid}, \mathsf{uid}, \mathsf{ctx}, \mathsf{sid})$

Require $(\cdot, \cdot, sid) \notin \mathsf{Q}_\tau$
$\tau \leftarrow \mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid, ctx, sid)$
If $(\tau \neq \bot)$ then
  $\mathsf{Q}_\tau := \mathsf{Q}_\tau \cup (uid, ctx, sid)$ // *Req. for Session Binding*
  If $(rid, crid, \cdot) \in \mathsf{Q}_{\mathsf{rid}}$ then // *Req. for Unlinkability*
    $\mathsf{Q}_{\mathsf{ppid}} := \mathsf{Q}_{\mathsf{ppid}} \cup \{(uid, rid)\}$
  Else $\mathsf{Q}_{\mathsf{ppid}} := \mathsf{Q}_{\mathsf{ppid}} \cup \{(uid, \mathsf{adv})\}$
Return τ

Oracle : $\mathsf{AResFin}(\mathsf{auth}, \mathsf{crid}, \mathsf{uid}, \mathsf{ctx}, \mathsf{sid})$

Require $(\cdot, \cdot, sid) \notin \mathsf{Q}_\tau \land (rid, crid, orid) \in \mathsf{Q}_{\mathsf{rid}}$
$\tau \leftarrow \mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid, ctx, sid)$
$(\tau_{\mathsf{fin}}, ppid) \leftarrow \mathsf{AFin}_{\mathsf{U}}(ipk, rid, crid, orid, ctx, sid, \tau)$
If $(\tau \neq \bot)$: $\mathsf{Q}_\tau := \mathsf{Q}_\tau \cup (uid, ctx, sid)$ // *Req. for Session Binding*
If $(\tau_{\mathsf{fin}}, ppid) \neq \bot$: $\mathsf{Q}_{\tau_{\mathsf{fin}}} := \mathsf{Q}_{\tau_{\mathsf{fin}}} \cup \{(rid, uid, ctx, sid)\}$
Return $(\tau_{\mathsf{fin}}, ppid)$

Oracle : $\mathsf{Vf}_{\mathsf{RP}}((\mathsf{rid}, \mathsf{ppid}, \mathsf{ctx}, \mathsf{sid}), \tau_{\mathsf{fin}})$

$b \leftarrow \mathsf{Vf}_{\mathsf{RP}}(ipk, (rid, ppid, ctx, sid), \tau_{\mathsf{fin}})$
If $(b = 1)$: $\mathsf{Q}_{\mathsf{vf}} := \mathsf{Q}_{\mathsf{vf}} \cup (rid, ppid, ctx, sid)$ // *Req. for Session Binding*
Return b

Figure 3: Definition of our security games and oracles used therein.

pseudonym and associated token trivial again. Thus, any implementation of our protocol must ensure that ctx does not disclose information that could link/identify users behind their pseudonyms.

3.2 Unobservability

This property captures that a malicious IdP does not learn anything about the RP's identity rid in an authentication request, meaning it cannot observe where the user wants to authenticate to. Specifically, the IdP should not be able to distinguish whether a user repeatedly authenticates to the same RP or different ones. This property was formally introduced in [32], and we simply adapt this to our notation. The game is represented as $\mathsf{Exp}^{\mathsf{UNOBS}}_{\mathcal{A},\mathsf{OPPID}}$ in Fig. 3. Unobservability is defined through an indistinguishability game, where the adversary, acting as a corrupt IdP, can set up RPs and obtain their authentication data via the corresponding oracles. Eventually the adversary chooses two RPs rid_0 and rid_1 and receives the authenticated request $auth_b, crid_b$ of either of them. The adversary wins if it can determine b better than by guessing.

Definition 3.2 (Unobservability). An OPPID scheme is unobservable if for all PPT adversaries $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{UNOBS}}_{\mathcal{A},\mathsf{OPPID}}(\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

3.3 Session Binding

The Session Binding property ensures that despite the privacy-preserving computation of an authentication token and pseudonym, the content of $(rid, ppid, ctx, sid)$ in the token is strictly unforgeable and pseudonyms are correctly formed. This comprises the classic unforgeability for all inputs directly seen and vouched for by the IdP

– which are (ctx, sid) for a *Direct Forgery* – but also all blindly signed information. The blindly signed information is $(rid, ppid)$, which the IdP vouches for in a session *sid* for user *uid* with context *ctx*. If the IdP indeed created a token for (ctx, sid), the blindly signed information must be consistent with its view and the intentions of all honest users and RPs. More precisely, the following must hold:

- If a user *uid* requested a token in session *sid*, it is infeasible to create a valid token for *rid*, *sid* and $ppid \neq \mathsf{F}(uid, rid)$ – *Nym Correctness.*
- If an honest user *uid* intended to authenticate to an RP *rid* in session *sid*, it must be infeasible to create a valid token for *sid* and another $rid' \neq rid$ – *RP Binding I.*
- If a corrupt user *uid* authenticated to an RP *rid* in a session *sid*, it is infeasible to generate valid authentication tokens for *sid* and more than *one* RP – *RP Binding II.*
- If an RP *rid* is not properly registered, it is infeasible to generate a valid token for *rid* – *RP Authentication I.*
- If an honest RP *rid* never authenticated for session *sid*, it is infeasible to create a valid token for *rid* – *RP Authentication II.*

We model the aforementioned properties through the game $\mathsf{Exp}^{\mathsf{SES\text{-}BIN}}_{\mathcal{A},\mathsf{OPPID}}$ (see Fig. 3), following the classic unforgeability setting. In this game, the IdP is honest, and the adversary can register both corrupt and honest RPs using oracles $\mathcal{O}$.RegHRP and $\mathcal{O}$.RegCRP, respectively, storing their registrations in HRP and CRP. The adversary initiates sessions for honest users via $\mathcal{O}.\mathsf{AInit}_{\mathsf{U}}$, obtains authentications from honest RPs via $\mathcal{O}.\mathsf{AReq}_{\mathsf{RP}}$, and acquires tokens from the IdP using $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ (for corrupt users) and $\mathcal{O}$.AResFin (for honest users). Additionally, we utilize a verification oracle $\mathcal{O}.\mathsf{Vf}_{\mathsf{RP}}$ to detect if the adversary attempts to reuse the same token across multiple (possibly corrupt) RPs.

The adversary can interact arbitrarily with these oracles and must output a token τ^*_{fin} for session $(rid^*, ppid^*, ctx^*, sid^*)$ as a forgery. The adversary wins the game if τ^*_{fin} is valid under *ipk* and the forgery is non-trivial, meaning it breaks any of the guarantees listed above, which are captured through dedicated winning conditions.

Direct and Indirect Forgeries. First, note that whenever the honest IdP creates a token for a session identified through (uid, ctx, sid), these values are stored in Q_τ. Thus, in the game, we check if the forgery is for $(\cdot, ctx^*, sid^*) \notin \mathsf{Q}_\tau$. If this occurs, $\mathcal{A}$ has produced a token for a session never attested by the honest IdP and wins under condition 1 (*Direct Forgery*).

The second category (*Indirect Forgery*) implies that the IdP did sign (ctx^*, sid^*), but the blindly signed or derived values $rid^*, ppid^*$ are inconsistent with the behavior of the other (honest) parties. This is captured under winning condition 2 and branches according to the properties we discussed earlier. In the following, we focus on Nym Correctness and refer to App. A for a detailed explanation of the RP Binding and Authentication properties.

Nym Correctness. Despite the blind pseudonym computation, a corrupt user *uid* must not be able to derive a token for any pseudonym other than the one uniquely defined through $\mathsf{F}(uid, rid)$, where *rid* is the RP specified in the token. This is captured in Condition (a), where $\mathcal{A}$ wins if $ppid^* \neq \mathsf{F}(uid, rid^*)$. This condition leverages the fact that the IdP receives *uid* as input, and we store *sid*, *uid* in Q_τ whenever a token is generated. Therefore, when the adversary outputs $(rid^*, ppid^*, ctx^*, sid^*)$, we can look up *uid* in Q_τ for sid^* and verify the correctness of the pseudonym for uid, rid^*. Recall that we already required F to produce unique pseudonyms, so this precisely defines the one pseudonym that is valid here, and $\mathcal{A}$ wins if it can produce a valid token for any other pseudonym value.

Note that this property, together with the uniqueness requirement of F, ensures *sybil-resistance.* This prevents malicious users from exploiting pseudonymous authentication to create several identities towards a single RP, which was not guaranteed in [28].

RP Binding & Authentication. The winning condition (b) for RP Binding I exploits that we know the intended *rid* when a session sid^* is started by an honest user through $\mathcal{O}.\mathsf{AInit}_{\mathsf{U}}$. Thus, if the adversary outputs a token for any $rid^* \neq rid$ for such a session, it wins the game. For a session sid^* initiated by a corrupt user towards a corrupt RP, we never know the exact RP the user wants to authenticate to: $\mathcal{A}$ invokes $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ with adversarially chosen inputs *auth* and *crid*, both of which hide *rid*. Thus, our guarantees are weaker here and follow the spirit of one-more unforgeability: the adversary wins if it has previously "presented" a valid token for some *rid* to the $\mathsf{Vf}_{\mathsf{RP}}$ oracle, yet later outputs a token for the same sid^* but with $rid^* \neq rid$ as a forgery. Catching such a "double spending" attack is the reason why we have the $\mathsf{Vf}_{\mathsf{RP}}$ oracle: it runs purely on public values, but essentially asks the adversary to commit to one view and later output a contradicting one as its forgery (see App. A for further explanation).

The preceding two properties ensure that the IdP-generated token is bound to the blindly received *rid*. Additionally, conditions (d) and (e) further ensure that only legitimate RPs can request such tokens. In condition (d), the adversary wins if it manages to produce a valid token for some rid^* that has never been registered, meaning $rid^* \notin \mathsf{HRP} \cup \mathsf{CRP}$. If the rid^* in the forgery belongs to an honest RP, we further let $\mathcal{A}$ win if $(rid^*, sid^*) \notin \mathsf{Q}_{\mathsf{auth}}$, indicating that the honest RP had never authenticated for that particular session sid^*.

Definition 3.3 (Session Binding). An OPPID scheme is Session Binding if for all PPT adversaries $\mathcal{A}$, $\Pr[\mathsf{Exp}^{\mathsf{SES\text{-}BIN}}_{\mathcal{A},\mathsf{OPPID}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Uniqueness of sid. We remark that our security notion relies on the fact that an honest IdP issues a single token per session *sid*. This ensures the desired freshness guarantees and uniquely identifies the session context that the IdP attests. As is typical in such protocols, we therefore assume that *sid* is unique per IdP and do not rely on the cryptographic protocol to enforce this. Thus, an implementation of OPPID must implement measures to ensure the freshness of *sid* at the application layer.

Weaknesses of [32]. Our notion builds upon the RP Session Binding model of [32]. Apart from adding Nym Correctness – which is the core functional extension needed for our work – our model significantly enhances the overall security guarantees provided by their notion. We address the following two weaknesses:

Firstly, the original model [32] only ensures security for sessions involving *honest* users. This restriction excludes scenarios involving "corrupt" user sessions, which are critical in real-world applications. The justification for this limitation lies in the blindness of *rid* towards the IdP: it is argued that the game requires the view of honest users to determine their targeted RP. However, this dependency is necessary only for security properties that depend on *rid*. Our security model carefully separates these dependencies into several sub-cases, where only RP Binding I necessitates the restriction to honest users. We demonstrate that even for corrupt users, a weaker form of *rid*-binding should be realized, as expressed in our RP Binding II condition.

Secondly, the original model captures security only against *corrupt* RPs, arguing that honest RPs do not give the adversary any advantage. However, higher security guarantees should ideally apply to sessions involving honest RPs as well. Specifically, the adversary should not be able to create any valid *rid*-bound token for sessions that the RP never authenticated, which we formalize in our RP Authentication II property. We stress that both are oversights in their security model only, as the protocol from [32] also satisfies our stronger security notion. However, related works such as PPOIDC [28] and UPPRESSO [27] do become insecure when users are corrupt, highlighting the need of a security model that properly captures malicious behavior. A more detailed comparison with the security model of [32] is provided in App. A.

3.4 Privacy Limitation: No Untraceability

While OPPID significantly enhances privacy in SSO, its guarantees are notably weaker compared to "full-fledged" privacy-preserving authentication systems. The primary limitation lies in the lack of *untraceability*, meaning OPPID does not provide privacy protection when the IdP and RP collude. We discuss these limitations here and argue that they are inherent in any SSO-like system.

When the IdP and (some) RPs collude, they can trace users through several means. First, through the deterministic pseudonyms, which is inherent in any pseudonymous SSO system where the only secret input to the pseudonym computation, $ppid = \mathsf{F}_k(uid, rid)$, is controlled by the IdP. As *uid* and *rid* are public information and typically stem from small "brute-forceable" sets, a colluding RP *rid* and IdP can determine the user behind an RP-specific pseudonyms through re-computation of the pseudonyms of all users and comparison against the *ppid* they want to identify. This is independent of *how* these pseudonyms are computed, and merely exploits their determinism. Possible means to mitigate that would be to distribute the IdP's key k among several IdPs or requiring some cryptographic input from the user. Both would deviate from core principles of SSO though, which relies on a single entity and does not assume the users to manage keys or credentials.

Second, a user's token request and the finalized token can be linked to each other, not only through the pseudonym, but also through the *sid* values known to both the IdP and RP, and through the timing between sessions that are handled simultaneously by both the IdP and RP. One can design a protocol where *sid* values are not revealed to the IdP in the clear, but the impact would be limited as the sessions can still be linked through the timing information. To avoid that linkage, one would have to break the immediate connection between the IdP and RP, e.g., by letting the IdP issue (somewhat) long-term credentials to users, and rely on techniques such as anonymous credentials for untraceable authentication from the user to the RP. In fact, this approach has already been proposed by EL PASSO [46], but gives up on the convenience advantage of plain SSO as it requires users to manage a long-term key.

4 OUR OPPID CONSTRUCTION

This section introduces our OPPID protocol π_{OPPID}, which combines oblivious *ppid* generation with a recent privacy-preserving RP authentication approach for the OIDC Implicit Flow [32]. We first outline the adapted authentication process from [32], present the construction of our pseudonym function and its semi-blind evaluation in the context of joint SSO authentication, and then proceed with the security analysis of our protocol. The detailed protocol, including oblivious *ppid* generation, is given in Fig. 4.

Our protocol uses signature schemes S_1 and S_2 with algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, where S_2 must support efficient NIZK proofs π of knowledge of a signature on committed values, using a commitment scheme $\mathsf{COM} = (\mathsf{Com}, \mathsf{Open})$. For pseudonyms, we need a group $\mathbb{G}$ of prime order q, a pseudorandom function $y \leftarrow \mathsf{PRF}(k, x)$ mapping to $\mathbb{Z}_q$, and a hash function H mapping to $\mathbb{G}$. Detailed definitions of these standard building blocks are provided in App. B.

4.1 Privacy-Preserving RP Authentication

Our protocol π_{OPPID} builds upon [32], which enables privacy-preserving RP authentication in SSO. The core idea therein is that an RP obtains a privacy-preserving credential from the IdP that includes the RP's identifier *rid*. This credential uses a signature scheme with efficient proofs, enabling the RP to authenticate to the IdP in a blind yet verifiable manner by sending a commitment to *rid* and proving ownership of a valid signature on *rid*. The IdP then verifies the proof and signs the commitment as part of the authentication token. Both the user and RP know the opening to the commitment and can verify that the token is indeed issued for the intended RP. The authentication token in [32] always contains the username *uid* that is vouched for by the IdP. In our protocol, the key modification is replacing *uid* with the pseudonym *ppid*, computed in a blind yet controlled way, detailed in Sec. 4.2.

The protocol in [32] also captures revocation by making the RP's credentials short-lived and encoding an epoch that must be revealed in every authentication. On the protocol level, adding epochs to credentials is very simple, but it makes the security model and analysis significantly more complex. Thus, we only use the core idea in our protocol and use it as a basis for integrating our privacy-preserving *ppid* generation. The main steps we use from [32] are related to setup, registration, and basic authentication.

Setup & Registration. The IdP generates key pairs for two signature schemes: $(sk_1, pk_1) \leftarrow_{\mathrm{R}} \mathsf{S}_1.\mathsf{KGen}(1^\lambda)$ for signing authentication tokens, and $(sk_2, pk_2) \leftarrow_{\mathrm{R}} \mathsf{S}_2.\mathsf{KGen}(1^\lambda)$ for efficient RP authentication proofs. During RP registration with identifier *rid*, the IdP issues a credential $cred := \sigma_{rid}$ for *rid*.

The public parameters $pp \leftarrow \mathsf{OPPID.Setup}(1^\lambda)$ include the descriptions of the underlying groups and the public parameters of COM, S_1, S_2, and NIZK. It is an implicit input for all algorithms.

Basic Authentication – From [32] Without Pseudonyms. For authentication of a user *uid* to an RP *rid* in a session identified through *sid*, the user, RP, and IdP proceed as follows:

(1) Initialization: The user creates a commitment *com* on the intended RP's *rid*, sends *com* and the opening o to the RP, and retains all values in its implicit state for later finalization.

(2) RP Authentication: When the RP receives a well-formed *com* and *o* for its *rid*, it generates its authentication *auth* by proving possession of the IdP's S_2 signature σ_{rid} on *rid* and that *rid* is included in *com*. The RP sends the proof π to the user, who forwards it along with the commitment *com* to the IdP.
(3) Token Generation: When the IdP receives the commitment *com* and the proof π from the user *uid*, it verifies the proof to ensure the authentication request is from a registered RP. If successful, the IdP generates a standard S_1 signature σ_τ on the commitment *com*, the session *sid*, and the context *ctx*.
(4) Finalization & Verification: To finalize the token τ, the user opens the commitment *com* with *o* and the intended *rid*, then verifies the IdP's signature. If successful, the opening *o* is added to τ, creating a verifiable binding between the IdP's signature and the specific RP.

Our construction lets the user create the commitment *com* and the opening *o*, but both could also be created by the RP and verified by the user. The two approaches do not differ in security they achieve, but we chose the former approach as it slightly simplified the security model. Furthermore, the commitment *com* and the blinded value $\overline{x}$ are verified twice to avoid stateful algorithm syntax and security model. If the user and RP can securely store verified information along with the *sid*, the repeated verification can be skipped.

4.2 Oblivious PPID Generation

The protocol outlined above would not be very useful yet, as it does not include a user identifier, which was simply *uid* in [32]. We now want to include a pseudonym $ppid = \mathsf{F}(uid, rid)$ in every token, where *uid* is the known user that the IdP has authenticated (outside of our protocol), and *rid* is the RP the user wants to authenticate to, but which must not be revealed to the IdP. We first describe the core function F and then explain how to compute it in a semi-blind and controlled manner, as required by our model.

Pseudonym Function F. Our pseudonym function F is a keyed deterministic function, which combines the standard DL-based pseudorandom function $\mathsf{F}_{\mathsf{DL}}(k, x) := \mathsf{H}(x)^k$ [36], operating in a group $\mathbb{G}$ of prime order q, and a hash function $\mathsf{H} : \mathcal{R} \mapsto \mathbb{G} \setminus \{1\}$, with a standard $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{U} \mapsto \mathbb{Z}_q$. The sets $\mathcal{U}$ and $\mathcal{R}$ represent the user and RP space, respectively. For compactness, we will sometimes write $\mathsf{F}_\mathsf{k}(uid, rid)$ to refer to this keyed function:

$$\mathsf{F}_{\mathsf{DL+PRF}} = \mathsf{F}_\mathsf{k}(uid, rid) := \mathsf{H}(rid)^{\mathsf{PRF}(k, uid)}.$$

The desired pseudonym unlinkability is directly ensured as $\mathsf{F}_{\mathsf{DL+PRF}}$ is a secure PRF. In fact, F_{DL} is often used to derive so-called scope-exclusive pseudonyms in the context of anonymous credentials, group signatures and DAA [9, 17, 35]. In these works, the exponent is a user-managed secret key, while we rely on the IdP to maintain them. Note that UPPRESSO [27] employs a similar keyed pseudonym function, and we refer for the comparison to Sec. 5.

Partially-Blind Evaluation of F. The construction $\mathsf{F}_{\mathsf{DL+PRF}}$ has also been used as a *partially-blind OPRF* in prior works [15, 31, 34], providing the capability for blind evaluation of the function on a hidden *rid* and a revealed *uid*. The user blinds the inner hash as $\overline{x} \leftarrow \mathsf{H}(rid)^r$ for a random r and sends this blinded value along with *uid* to the IdP. The IdP responds with $\overline{y} \leftarrow \overline{x}^{\mathsf{PRF}(k, uid)}$, where the exponent depends on the revealed *uid*. The user then unblinds the response to $ppid \leftarrow \overline{y}^{-r}$ to obtain the expected pseudonym:

$$ppid = \overline{y}^{-r} = \left((\mathsf{H}(rid)^r)^{\mathsf{PRF}(k, uid)} \right)^{-r} = \mathsf{H}(rid)^{\mathsf{PRF}(k, uid)}.$$

The Need for Verifiability. Finally, another essential feature required for our function is verifiability: the RP must be assured that a received *ppid* was computed for the correct *rid*. While the IdP is trusted here, one approach could be to delegate this verification task to the IdP by employing a partially-blind OPRF with *committed and verifiable inputs*. This approach would enable the IdP to verify that it evaluates the blind function on the same *rid* that is authenticated through *auth* (and contained in *com*). Such a function would serve as a suitable building block, but it is not known whether such an OPRF is feasible [14]. Existing constructions like the Dodis-Yampolskiy (O)PRF [18], which operate on homomorphically encrypted inputs, allow for proofs of well-formedness but do not extend to the partially-blind setting needed here. Similarly, constructions like (2)HashDH, which enable partial blindness, lack efficient and composable proofs of correct inputs as their input is a perfectly blinded hash value that destroys all algebraic structure.

$\mathsf{KGen}_{\mathsf{IdP}}(\mathsf{pp}) \rightarrow ((\mathsf{isk}, \mathcal{M}), \mathsf{ipk})$

$(sk_1, pk_1) \leftarrow_{\mathrm{R}} \mathsf{S}_1.\mathsf{KGen}(1^\lambda)$; $(sk_2, pk_2) \leftarrow_{\mathrm{R}} \mathsf{S}_2.\mathsf{KGen}(1^\lambda)$; $k \leftarrow_{\mathrm{R}} \{0,1\}^\lambda$
Return $(((sk_1, sk_2, k), \emptyset), (pk_1, pk_2))$

$\langle \mathsf{Join}_{\mathsf{RP}}(\mathsf{ipk}, \mathsf{rid}), \mathsf{Reg}_{\mathsf{IdP}}(\mathsf{isk}, \mathsf{rid}, \mathcal{M}) \rangle \rightarrow \{(\mathsf{cred}, \mathcal{M}'), \bot\}$

RP : Initiate registration for rid
IdP : Parse isk as $(\cdot, sk_2, \cdot)$; Require $rid \notin \mathcal{M}$
$\sigma_{rid} \leftarrow \mathsf{S}_2.\mathsf{Sign}(sk_2, rid)$; $\mathcal{M}' \leftarrow \mathcal{M} \cup \{rid\}$
RP : Return σ_{rid} ; IdP : Return $\mathcal{M}'$

$\mathsf{AInit}_{\mathsf{U}}(\mathsf{ipk}, \mathsf{rid}) \rightarrow (\mathsf{orid}, \mathsf{crid})$

$(com, o) \leftarrow_{\mathrm{R}} \mathsf{Com}(rid)$; $r \leftarrow_{\mathrm{R}} \mathbb{Z}_q^*$; $\overline{x} \leftarrow \mathsf{H}(rid)^r$
Return $((r, o), (\overline{x}, com))$

$\mathsf{AReq}_{\mathsf{RP}}(\mathsf{ipk}, \mathsf{rid}, \mathsf{cred}, \mathsf{crid}, \mathsf{orid}, \mathsf{sid}) \rightarrow \mathsf{auth}$

Parse $cred$ as σ_{rid}, ipk as $(\cdot, pk_2)$, $crid$ as $(\overline{x}, com)$, $orid$ as (r, o)
Require $\mathsf{H}(rid)^r = \overline{x} \wedge \mathsf{Open}(rid, com, o) = 1$
Return $\pi \leftarrow \mathsf{NIZK}\{(rid, o, \sigma_{rid}) : \mathsf{S}_2.\mathsf{Vf}(pk_2, rid, \sigma_{rid}) = 1 \wedge \mathsf{Open}(rid, com, o) = 1\}(com, \overline{x}, sid)$

$\mathsf{ARes}_{\mathsf{IdP}}(\mathsf{isk}, \mathsf{auth}, \mathsf{crid}, \mathsf{uid}, \mathsf{ctx}, \mathsf{sid}) \rightarrow \{\tau, \bot\}$

Parse ipk as $(\cdot, pk_2)$, isk as $(sk_1, \cdot, k)$, $crid$ as $(\overline{x}, com)$, $auth$ as π
Require that π verifies w.r.t. $(pk_2, com, \overline{x}, sid)$ and $\overline{x} \in \mathbb{G}$
$uk \leftarrow \mathsf{PRF}(k, uid)$; $\overline{y} \leftarrow \overline{x}^{uk}$
$\sigma_\tau \leftarrow \mathsf{S}_1.\mathsf{Sign}(sk_1, (com||\overline{x}||\overline{y}||ctx||sid))$; Return $(\sigma_\tau, \overline{y})$

$\mathsf{AFin}_{\mathsf{U}}(\mathsf{ipk}, \mathsf{rid}, \mathsf{crid}, \mathsf{orid}, \mathsf{ctx}, \mathsf{sid}, \tau) \rightarrow \{(\tau_{\mathsf{fin}}, \mathsf{ppid}), \bot\}$

Parse ipk as $(pk_1, \cdot)$, $crid$ as $(\cdot, com)$, $orid$ as (r, o), τ as $(\sigma_\tau, \overline{y})$
$\overline{x} \leftarrow \mathsf{H}(rid)^r$; $y \leftarrow \overline{y}^{-r}$
Require $\mathsf{Open}(rid, com, o) = \mathsf{S}_1.\mathsf{Vf}(pk_1, (com||\overline{x}||\overline{y}||ctx||sid), \sigma_\tau) = 1$
Return $((com, o, r, \overline{y}, \sigma_\tau), y)$

$\mathsf{Vf}_{\mathsf{RP}}(\mathsf{ipk}, (\mathsf{rid}, \mathsf{ppid}, \mathsf{ctx}, \mathsf{sid}), \tau_{\mathsf{fin}}) \rightarrow 0/1$

Parse ipk as $(pk_1, \cdot)$, τ_{fin} as $(com, o, r, \overline{y}, \sigma_\tau)$
$\overline{x} \leftarrow \mathsf{H}(rid)^r$; $y \leftarrow \overline{y}^{-r}$
Return 1 if
$\mathsf{Open}(rid, com, o) = \mathsf{S}_1.\mathsf{Vf}(pk_1, (com||\overline{x}||\overline{y}||ctx||sid), \sigma_\tau) = 1 \wedge ppid = y$

Figure 4: π_{OPPID} protocol construction of our OPPID system.

Adding Partial Verifiability. Interestingly, we can work around this non-existent building block by incorporating several straightforward steps, building upon UPPRESSO [27] and the base protocol from [32]. The resulting protocol is detailed in Fig. 4.

First, in addition to the commitment and opening com, o for rid, we let the user compute $\overline{x} = \mathsf{H}(rid)^r$ and send all values, including r and o, to the RP. The RP verifies that both $\overline{x}$ and com open to its rid. Only upon successful verification does the RP provide its authentication $auth$, proving ownership of a valid credential for the committed rid in com. The RP also binds its NIZK proof π to $\overline{x}$. This ensures $\overline{x}$ correctness when either the RP or user is honest.

Second, upon receiving a verified authentication request, the IdP includes both the blinded input $\overline{x}$ and the blinded output $\overline{y}$ in its token, as done in [27], signing $\sigma_\tau \leftarrow \mathsf{S}_1.\mathsf{Sign}(sk_1, (com||\overline{x}||\overline{y}||ctx||sid))$. Crucially, the signature binds the blinded and non-verified values $\overline{x}$ and $\overline{y}$ used for $ppid$ to the commitment com on rid, for which the RP provided a valid NIZK proof.

Third, the user incorporates the blinding value r used to hide rid in $\overline{x}$ as part of the final token τ_{fin}. Thus, $\tau_{\mathsf{fin}} = (com, o, r, \overline{y}, \sigma_\tau)$, and the verification function $\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid, ppid, ctx, sid), \tau_{\mathsf{fin}})$ performs the following crucial checks:

- Verify that the IdP's signature σ_τ is valid for the recomputed $\overline{x} = \mathsf{H}(rid)^r$, where rid is the one provided in the verification.
- Ensure $ppid$ satisfies $ppid = \overline{y}^{\,-r}$, where $\overline{y}$ is signed by the IdP and r is the blinding value leading to the correct $\overline{x}$.
- Confirm that the rid contained in $\overline{x}$ matches the one in com.

These checks extend the guarantees from com to $ppid$ by leveraging our three-party setting, where both the user and RP know rid and verify that $\overline{x}$ and com are valid for rid. Thus, as long as either the RP or the user remains honest, the derived pseudonym $ppid = \mathsf{F}(uid, rid)$ is guaranteed to be correct. These checks also explain why the RP did not have to prove the well-formedness of $\overline{x}$ explicitly, but instead, including it in the ZKP hash was sufficient.

Invalid Pseudonyms – If Both RP and User Are Corrupt. If both the RP and user are corrupt, they have some leeway, but none that is harmful. A malicious RP rid and a malicious user uid can request and obtain pseudonyms $ppid = \mathsf{F}(uid, rid')$ for arbitrary $rid' \neq rid$ by sending a blinded $\overline{x}$ to the IdP that does not contain the correct rid. The IdP will compute the pseudonym based on the incorrect rid' but will bind it to the verified commitment com, which can only be opened to rid'. In the final token, the rid is no longer blinded, and verification includes a check whether the commitment contains the same rid as the blinded $\overline{x}$ used in the $ppid$ computation. This check will fail, rendering the entire token and pseudonym invalid.

Furthermore, note that this "attack" is only feasible for malicious users uid, as uid is revealed to the IdP and used to compute $ppid$. Thus, a malicious RP and user uid cannot trick the IdP into computing pseudonyms for any other user $uid' \neq uid$.

4.3 Security Analysis

We have already informally sketched how the different security properties are guaranteed in our protocol. Now, we formally prove that our protocol π_{OPPID} satisfies all security and privacy properties defined in Sec. 3. Note that the proof of Request Authentication essentially follows the one in [32] and can thus be found in App. C.4. Furthermore, the properties of our pseudonym function $\mathsf{F}_{\mathsf{DL+PRF}}$ – uniqueness, collision-freeness, and unlinkability – follow from the security of the underlying PRF and HashDH. We elaborate further on these properties in App. C.

Analysis of π_{OPPID}. We now turn to the proofs of our three core properties, ensuring the correct yet privacy-preserving computation of pseudonymous authentication tokens with respect to our pseudonym function $\mathsf{F}_{\mathsf{DL+PRF}}$.

Theorem 4.1 (Unlinkability). *π_{OPPID} satisfies Unlinkability if* H *is a random oracle,* PRF *is a secure pseudorandom function and the DDH assumption holds in* $\mathbb{G}$.

This proof relies on the pseudorandomness of $\mathsf{F}_{\mathsf{DL+PRF}}$, as shown in App. C.2, under the assumptions that H is a random oracle, PRF is a secure pseudorandom function, and the DDH assumption holds in $\mathbb{G}$. We now provide a proof sketch and refer to App. C.1 for the full proof, where we also discuss why a one-more-type assumption, often required for OPRFs, is not required.

Proof sketch. In the Unlinkability game, the adversary aims to determine the user uid_b behind a pseudonym $ppid_b$ and token $\tau_{\mathsf{fin}\,b}$, generated for rid and either uid_0 or uid_1. It has oracle access to the honest IdP and can learn the pseudonyms of uid_0 and uid_1 for all RPs except rid (as otherwise winning is trivial).

We already know that $\mathsf{F}_{\mathsf{DL+PRF}}$ is a secure PRF, meaning the $ppid$s themselves do not leak any information about the contained uid, except what is deterministically derived. The only part in the IdP's response τ_{fin} that depends on uid is the PRF output $\overline{y} = \overline{x}^{\mathsf{PRF}(k,uid)}$, where $\overline{x}$ is the value $\mathsf{F}_{\mathsf{k}}(uid, rid) = \mathsf{H}(rid)^{\mathsf{PRF}(k,uid)}$ blinded with a random r. Note that Unlinkability holds for honest users only, ensuring that $\overline{x} = \mathsf{H}(rid)^r$ in the challenge query is a valid input. Thus, the token does not provide the adversary with any information beyond $ppid$.

What remains to be shown is that a malicious RP, possibly colluding with a malicious user uid^*, cannot exploit the partially blind evaluation of $\mathsf{F}_{\mathsf{DL+PRF}}$ to obtain dedicated $ppid$s of either of the honest challenge users uid_0 or uid_1 illegitimately. Specifically, they cannot obtain their pseudonyms through oracle queries *not* intended for either uid_0 or uid_1 (as for the challenge users, the oracles enforce honest user behavior and honestly generated inputs $\overline{x}$).

It is easy to see that this scenario is infeasible because the $ppid$ depends on the uid, which the IdP learns in clear and uses in its computation. Therefore, there is no opportunity to manipulate the uid and its impact on the $ppid$ computation. While a malicious RP and user could potentially trick the IdP into computing a pseudonym for an arbitrary rid that does not match the one authenticated via $auth$, they can only do so for a malicious $uid^* \neq uid_0, uid_1$, which does not provide any advantage in winning the Unlinkability game. □

Theorem 4.2 (Unobservability). *π_{OPPID} satisfies Unobservability if* COM *is hiding, and the* NIZK *is zero-knowledge.*

This proof is essentially the same as in [32]. It follows from the fact that the IdP receives the rid in a commitment and within a zero-knowledge proof. The only difference here is that $\mathcal{A}$ also receives $\overline{x} = \mathsf{H}(rid)^r$, which is the blinded hash of rid. As the blinding is information-theoretic, no additional assumptions are needed. We provide a simple proof in App. C.2 for completeness.

Theorem 4.3 (Session Binding). π_{OPPID} *satisfies Session Binding if the* S_1 *and* S_2 *scheme are EUF-CMA secure,* COM *is binding, and the* NIZK *is zero-knowledge and simulation extractable.*

We now sketch the proof and give a full proof in App. C.3.

Proof sketch. The proof branches along the winning conditions in the Session Binding game, where the cases for *Direct Forgery*, RP Binding and RP Authentication are handled analogously to [32], as these properties mostly follow from their protocol we have used as basis for our extension. Thus, we focus on Nym Correctness (Case 2a) here: it must be infeasible to output $(rid^*, ppid^*, ctx^*, sid^*, \tau^*_{\mathsf{fin}})$ where τ^*_{fin} is a valid token for a pseudonym $ppid^* \neq \mathsf{F}(uid, rid^*)$. That is, the pseudonym in the forgery does not match $\mathsf{F}(uid, rid^*)$ which is supposed to be unique. Recall that the final token τ^*_{fin} contains $(o^*, r^*, \overline{y}^*, \sigma^*_\tau)$, and as τ^*_{fin} is valid, $ppid^* = \overline{y}^{*-r^*}$ and σ^*_τ is a valid signature on $(com^*||\overline{x}^*||\overline{y}^*)$. These values uniquely determine the correct pseudonym value for which verification succeeds as verification checks that $\overline{x}^* = H(rid^*)^{r^*}$ and $ppid^* = \overline{y}^{*-r^*}$.

Thus, the only way to pass the verification check and output a pseudonym $ppid^* \neq \mathsf{F}(uid, rid^*)$ is changing the signed $\overline{x}^*, \overline{y}^*$ values in the token. This is infeasible if S_1 is unforgeable. □

5 EVALUATION & DISCUSSION

This section presents our prototypical implementation of π_{OPPID} and compares its efficiency and security to the most related work.

5.1 Security Comparison with Related Protocols

The works most closely related to ours are standard OIDC with pseudonyms [43], AIF-ZKP [32], PPOIDC [28], and UPPRESSO [27]. These protocols have been selected for a detailed comparison as they also operate within the plain-SSO model, meaning they do not require the user to manage any long-term keys or credentials, nor do they rely on additional parties or dedicated hardware modules. For space reasons, the straightforward comparison to OIDC and AIF-ZKP, both of which entirely lack either Unlinkability or Unobservability, is delegated to App. C.5.

PPOIDC [28]. The PPOIDC protocol aims to turn OIDC into an unobservable protocol. To blindly bind the IdP's token to a particular rid and compute the $ppid$ it mostly relies on hash functions, serving as commitments. More precisely, when the user wants to authenticate to rid, it first computes $com_{rid} := \mathsf{H}(rid||r)$ for a random $r \leftarrow_{\mathrm{R}} \{0,1\}^\lambda$. The pseudonym computation uses a hash function again and also makes the non-standard assumption that uid is a high-entropy value that the user retrieves from the IdP at every login. The user computes $ppid := \mathsf{H}(uid||rid)$ and the commitment $com_{ppid} := \mathsf{H}(ppid||r')$ with $r' \leftarrow_{\mathrm{R}} \{0,1\}^\lambda$. The user also generates a zero-knowledge proof π that com_{ppid} is derived for her uid and sends com_{rid}, com_{ppid} and π to the IdP.

The IdP verifies that the proof is valid for the authenticated uid, and then signs com_{rid} and com_{ppid} in its token. The user forwards the IdP's signature, randomness r, r' and her $ppid$ to the RP, which checks that the IdP-signed commitment com_{rid} correctly opens to its own rid and com_{ppid} opens to $ppid$.

The protocol does not consider RP authentication towards the IdP and thus cannot satisfy these parts of the Session Binding property or Request Authentication. RP authentication is not entirely missing either though, as their protocol issues certificates to the RP upon registration and relies on the user to verify them when they start a session. This only provides security if all users are honest though, and we discuss the difference to our IdP-centric authentication at the end of this section. The PPOIDC still partially satisfies Session Binding, as the IdP blindly signs the commitment com_{rid} of an user-verified rid, which ensures RP Binding I and II.

However, PPOIDC lacks *Nym Correctness*: The proof π ensures that $ppid$ is computed on the correct uid but does not guarantee that the committed rid in com_{rid} matches the one used to compute $ppid$, allowing corrupt users to obtain arbitrary IdP-certified pseudonyms.

UPPRESSO [27]. The protocol focuses solely on the blindly computed pseudonyms in SSO, which are computed as $ppid := rid^{k_{uid}}$ where k_{uid} is a user-specific secret key in $\mathbb{Z}_q$ maintained by the IdP. Instead of hashing rid to the group as in π_{OPPID}, their protocol relies on rid already being a proper and random group element. This is done by letting the IdP issue a certificate on a randomly chosen group element $rid \in \mathbb{G}$ (where $\mathbb{G}$ is a cyclic group of order q) to the RP when it registers.

When a user wants to authenticate to an RP, it receives and verifies the certified rid from the RP and uses a random r to blind it, $\overline{rid} := rid^r$. The IdP then receives $\overline{rid}$ from the user uid, computes $\overline{ppid} := \overline{rid}^{k_{uid}}$, and signs both $\overline{rid}$ and $\overline{ppid}$ in its token. The RP then receives all signed values and r and unblinds them to $ppid := rid^{k_{uid}}$.

In terms of security, UPPRESSO achieves both privacy-related properties due to perfect blinding of rid and pseudonyms computed via a classic DL-based PRF. The protocol partially achieves RP-Binding, as the signed $\overline{rid}$ could serve as a commitment to rid which can be verified using r (this is not made publicly verifiable in their protocol though). In contrast to our scheme, this would additionally require to also verify that rid is the correct group element – which was simply computing $\mathsf{H}(rid)$ in our scheme. However, UPPRESSO does not achieve RP authentication as part of Session Binding or Request Authentication, as the RP does not authenticate to the IdP.

Comparing UPPRESSO to our protocol, we made three key changes to the pseudonym computation: First, we set $k_{uid} := \mathsf{PRF}(k, uid)$, where k is a secret key held by the IdP, avoiding the need for the IdP to manage a secret key table that grows linearly with the number of users. Second, we compute pseudonyms on $\mathsf{H}(rid)$ instead of a *certified* $rid \in \mathbb{G}$ directly, removing the need for users to verify a certificate on rid to check its validity. Using a malformed rid will allow malicious RPs to link users, which is prevented through our hash computation. Third, our protocol ensures that valid, publicly verifiable IdP tokens include the correct, RP-authenticated pseudonym by enforcing consistency checks between the verified commitment and blinded input. In UPPRESSO, the final token can include malformed pseudonyms when corrupt users and RPs collude, allowing the RP to falsely claim an inflated user base asserted by the IdP. Other changes primarily focus on providing IdP-side RP authentication and ensuring the final token's public verifiability for the targeted rid.

We note that the recently proposed BISON protocol [29] for blindly computed SSO pseudonyms essentially follows the same approach as UPPRESSO. In particular, it does not aim at RP authentication towards the IdP, and thus our analysis and comparison for UPPRESSO also carry over to BISON.

User-Side RP Authentication. In PPOIDC and UPPRESSO, RP authentication is shifted from an IdP-verified setting to a user-verified one. In this approach, the user receives an RP's certificate to verify that it is properly registered with the IdP and provides the correctly certified values for the cryptographic protocol. UPPRESSO requires two RP and IdP scripts that the user receives from each party, which handle certificate transfer and verification within the session.

Interestingly, both protocols suggest sending the plain RP certificate to the user without binding the certificate to a key and session nonce. This approach could enable phishing attacks if no additional cross-verification of consistent certificates from the authenticated TLS session and the verified SSO-protocol values is performed.

Furthermore, relying on proper RP authentication by the user compromises full IdP control over the token it issues. As discussed above, the IdP can be tricked into signing tokens for malformed pseudonyms or invalid *rid*s, violating the correctness and non-repudiation guarantees typically expected from such an IdP. Overall, handling RP authentication on the user side is rather fragile and requires trust in the honest execution on the user's device. Therefore, our OPPID system aims for IdP-side RP authentication.

RP Revocation. Achieving Unobservability, which hides *rid* from the IdP during RP authentication, rules out classic revocation strategies based on blacklisting revoked *rid*s. This well-understood challenge is addressed by [32] through an epoch-based strategy, where RP credentials used in anonymous authentication requests are short-lived (e.g., a week or a month). Thus, the RP must regularly re-obtain its membership credential from the IdP, which will refuse to do so if the RP has been revoked. The short-lived credentials can be realized by having the IdP also sign the current epoch along with the RP's *rid* in the membership credential. During authentication, the *rid* remains hidden, but the epoch must be revealed and valid.

Since π_{OPPID} is built upon [32], integrating this revocation mechanism only requires adding an epoch to the issued RP credential. This is straightforward and has no impact on how the *ppid* is generated or verified. We omitted revocation to simplify the security model and focus on oblivious *ppid* computation.

5.2 Implementation and Evaluation

We now report on the implementation of our protocol and compare its efficiency to related works.

Instantiation of Building Blocks. We instantiated π_{OPPID} scheme as follows: for the IdP's standard signature S_1, we used RSA-SHA256 2048-bit to comply with current industry standards. Building upon [32], we used PS signatures [41] on curve BLS12-381 [5] for S_2, which includes the bilinear group description $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ in the public parameters *pp*. The commitment is instantiated with Pedersen commitments [39] in $\mathbb{G}_1$, and the pseudonym function is executed in $\mathbb{G}_1$ as well, with hashing to the curve [21] and the "exponent" PRF realized with HMAC-SHA256 mapping elements to $\mathbb{Z}_q$ [21, §5.3]. The NIZK instantiation, used in both π_{OPPID} and AIF-ZKP for RP authentication, uses generalized Schnorr-proofs [11], made non-interactive through the Fiat-Shamir heuristic [22].

Entity	RP		User		IdP
Protocol \ Ops.	*Vf*	*Req*	*Fin*	*Init*	*Res*
OIDC [43]	0.05	n/a	n/a	n/a	1.52
AIF-ZKP [32]	1.12	7.83	1.11	1.19	12.09
PPOIDC [28]	0.06	n/a	n/a	4710	5.88
UPPRESSO [27]	0.57	0.44	n/a	0.52	1.98
Our Work: π_{OPPID}	2.32	8.47	2.28	1.81	12.79

Table 2: Execution times (in ms), including proof generation and verification when applicable. For an overview of the achieved security and privacy properties, see Tab. 1.

For comparison with OIDC, PPOIDC, and UPPRESSO, we used the same RSA signature for the IdP-signed token. For OIDC, we followed the specification [43, §8.1] and used SHA256 for the *ppid* computation. Since no implementation was available for PPOIDC, we implemented their scheme from scratch, using SHA256 for H to instantiate $\mathsf{F}(uid, rid)$ and the commitment scheme as proposed in [28]. For the ZKP π of a pre-image of H, we used *gnark* [7] to create the circuit-based zkSNARK, which is verified by the IdP. For UPPRESSO, we implemented its core cryptographic operations: IdP-issued RP certificates are realized via an RSA-SHA256 signature on *rid* only and pseudonyms are computed in $\mathbb{G}_1$.

Evaluation Results. The benchmark results of all schemes are summarized in Tab. 2, with all operations performed on an Apple M1 CPU (8-core, 2020, 3.2 GHz). Our implementation and benchmarks are available at [3] for reproducibility.

Our protocol π_{OPPID}, achieving all desired security and privacy properties, is highly efficient. User operations and token verification each take only 2ms, proof verification at the IdP requires only 12ms after a 8ms generation time at the RP. We now discuss these results in relation to the closest variants, PPOIDC and UPPRESSO.

Our scheme is significantly faster than PPOIDC, which requires an expensive proof generation of 4.7s by the user, which came for a fast IdP verification of only 5ms. One might speed up the proof generation of a hash preimage by using alternative zkSNARK setups, but that would increase verification or communication costs [42]. UPPRESSO has almost an identical pseudonym computation as in π_{OPPID} and does not involve the RP authentication costs, so it is slightly faster than ours, but provides less security.

Regarding communication costs, an element in $(\mathbb{Z}_q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ requires $(32, 48, 92, 576)$ bytes respectively. Therefore, π_{OPPID} requires only 864 bytes $(3\mathbb{Z}_q + 4\mathbb{G}_1 + 1\mathbb{G}_T)$ for the authentication proof and the blinded and committed *rid* sent to the IdP. The user-generated ZKP in PPOIDC is relatively small at 997 bytes, but the associated costs are substantial due to the pre-compiled circuit (309407 constraints, 56 MB) and a 121 MB proving key. Fetching these files from the IdP for each login is clearly impractical. The overhead introduced by user-side RP certificates in PPOIDC/UPPRESSO is relatively minor at 32 bytes for the signature in addition to a signed 4-byte *rid*. Thus, in scenarios where RP authentication would *not* be needed, our protocol remains more efficient than PPOIDC.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback throughout the review process, as well as Christian Mouchet for his artifact review. This research was partially funded by the HPI Research School on Data Science and Engineering. It was also supported by the German Federal Ministry of Education and Research (BMBF) through funding of the ATLAS project under reference number 16KISA037.

REFERENCES

[1] Google 2023. *OpenID Connect.* Google. https://developers.google.com/identity/protocols/oauth2/openid-connect
[2] Apple Inc. 2023. *Sign in with Apple.* Apple Inc. https://developer.apple.com/documentation/sign_in_with_apple
[3] 2024. *OPPID.* https://github.com/jmakr0/OPPID
[4] Apple. 2023. *Sign in with Apple & Privacy.* https://www.apple.com/legal/privacy/data/en/sign-in-with-apple/
[5] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. 2003. Constructing Elliptic Curves with Prescribed Embedding Degrees. In *SCN 02 (LNCS, Vol. 2576)*, Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano (Eds.). Springer, Heidelberg, Germany, Amalfi, Italy, 257–267. https://doi.org/10.1007/3-540-36413-7_19
[6] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *20th ACM STOC*. ACM Press, Chicago, IL, USA, 103–112. https://doi.org/10.1145/62212.62222
[7] Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. 2023. *ConsenSys/gnark: v0.10.0.* https://doi.org/10.5281/zenodo.5819104
[8] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In *ACM CCS 2004*, Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel (Eds.). ACM Press, Washington, DC, USA, 132–145. https://doi.org/10.1145/1030083.1030103
[9] Jan Camenisch, Manu Drijvers, and Anja Lehmann. 2016. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. In *Trust and Trustworthy Computing*, Michael Franz and Panos Papadimitratos (Eds.). Springer International Publishing, Cham, 1–20.
[10] Jan Camenisch, Maria Dubovitskaya, Robert R. Enderlein, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. 2014. Concepts and languages for privacy-preserving attribute-based authentication. *J. Inf. Secur. Appl.* 19, 1 (Feb. 2014), 25–44. https://doi.org/10.1016/j.jisa.2014.03.004
[11] Jan Camenisch, Aggelos Kiayias, and Moti Yung. 2009. On the Portability of Generalized Schnorr Proofs. In *EUROCRYPT 2009 (LNCS, Vol. 5479)*, Antoine Joux (Ed.). Springer, Heidelberg, Germany, Cologne, Germany, 425–442. https://doi.org/10.1007/978-3-642-01001-9_25
[12] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001 (LNCS, Vol. 2045)*, Birgit Pfitzmann (Ed.). Springer, Heidelberg, Germany, Innsbruck, Austria, 93–118. https://doi.org/10.1007/3-540-44987-6_7
[13] Jan Camenisch and Els Van Herreweghen. 2002. Design and Implementation of The Idemix Anonymous Credential System. In *ACM CCS 2002*, Vijayalakshmi Atluri (Ed.). ACM Press, Washington, DC, USA, 21–30. https://doi.org/10.1145/586110.586114
[14] Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. 2022. SoK: Oblivious Pseudorandom Functions. Cryptology ePrint Archive, Report 2022/302. https://eprint.iacr.org/2022/302.
[15] Poulami Das, Julia Hesse, and Anja Lehmann. 2022. DPaSE: Distributed Password-Authenticated Symmetric-Key Encryption, or How to Get Many Keys from One Password. In *ASIACCS 22*, Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (Eds.). ACM Press, Nagasaki, Japan, 682–696. https://doi.org/10.1145/3488932.3517389
[16] Arkajit Dey and Stephen Weis. 2010. PseudoID: Enhancing Privacy in Federated Login. In *Hot Topics in Privacy Enhancing Technologies*. Sciendo, Berlin, Germany, 95–107.
[17] Jesus Diaz and Anja Lehmann. 2021. Group Signatures with User-Controlled and Sequential Linkability. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, Germany, Virtual Event, 360–388. https://doi.org/10.1007/978-3-030-75245-3_14
[18] Yevgeniy Dodis and Aleksandr Yampolskiy. 2005. A Verifiable Random Function with Short Proofs and Keys. In *PKC 2005 (LNCS, Vol. 3386)*, Serge Vaudenay (Ed.). Springer, Heidelberg, Germany, Les Diablerets, Switzerland, 416–431. https://doi.org/10.1007/978-3-540-30580-4_28
[19] EU. 2024. *Regulation 2024/1183 of the european parliament and of the council of 11 april 2024 amending regulation no 910/2014 as regards establishing the european digital identity framework.* https://eur-lex.europa.eu/eli/reg/2024/1183/oj
[20] Facebook. 2021. *OpenID Connect.* https://developers.facebook.com/docs/facebook-login
[21] A. Faz-Hernandez, S. Scott, N. Sullivan, R. S. Wahby, and C. A. Wood. 2023. *Hashing to Elliptic Curves.* RFC 9380. RFC Editor.
[22] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS, Vol. 263)*, Andrew M. Odlyzko (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 186–194. https://doi.org/10.1007/3-540-47721-7_12
[23] Ge Gao, Yuan Zhang, Yaqing Song, and Shiyu Li. 2024. PrivSSO: Practical Single-Sign-On Authentication Against Subscription/Access Pattern Leakage. *IEEE Transactions on Information Forensics and Security* 19 (2024), 5075–5089. https://doi.org/10.1109/TIFS.2024.3392533
[24] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. 1988. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. *SIAM J. Comput.* 17, 2 (April 1988), 281–308.
[25] Jens Groth. 2006. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT 2006 (LNCS, Vol. 4284)*, Xuejia Lai and Kefei Chen (Eds.). Springer, Heidelberg, Germany, Shanghai, China, 444–459. https://doi.org/10.1007/11935230_29
[26] Chengqian Guo, Fan Lang, Qiongxiao Wang, and Jingqiang Lin. 2022. UP-SSO: Enhancing the User Privacy of SSO by Integrating PPID and SGX. In *2021 International Conference on Advanced Computing and Endogenous Security.* 01–05. https://doi.org/10.1109/IEEECONF52377.2022.10013340
[27] Chengqian Guo, Jingqiang Lin, Quanwei Cai, Wei Wang, Fengjun Li, Qiongxiao Wang, Jiwu Jing, and Bin Zhao. 2022. UPPRESSO: Untraceable and Unlinkable Privacy-PREserving Single Sign-On Services. arXiv:2110.10396 [cs.CR] https://arxiv.org/abs/2110.10396
[28] Sven Hammann, Ralf Sasse, and David A. Basin. 2020. Privacy-Preserving OpenID Connect. In *ASIACCS 20*, Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese (Eds.). ACM Press, Taipei, Taiwan, 277–289. https://doi.org/10.1145/3320269.3384724
[29] Jakob Heher, Stefan More, and Lena Heimberger. 2024. BISON: Blind Identification with Stateless scOped pseudoNyms. arXiv:2406.01518 [cs.CR] https://arxiv.org/abs/2406.01518
[30] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. 2014. Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model. In *ASIACRYPT 2014, Part II (LNCS, Vol. 8874)*, Palash Sarkar and Tetsu Iwata (Eds.). Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C., 233–253. https://doi.org/10.1007/978-3-662-45608-8_13
[31] Stanislaw Jarecki, Hugo Krawczyk, and Jason Resch. 2018. Threshold Partially-Oblivious PRFs with Applications to Key Management. Cryptology ePrint Archive, Report 2018/733. https://eprint.iacr.org/2018/733.
[32] Maximilian Kroschewski and Anja Lehmann. 2023. Save The Implicit Flow? Enabling Privacy-Preserving RP Authentication in OpenID Connect. *Proceedings on Privacy Enhancing Technologies* 4 (2023), 96–116.
[33] Leona Lassak, Elleen Pan, Blase Ur, and Maximilian Golla. 2024. Why Aren't We Using Passkeys? Obstacles Companies Face Deploying {FIDO2} Passwordless Authentication. 7231–7248. https://www.usenix.org/conference/usenixsecurity24/presentation/lassak
[34] Anja Lehmann. 2019. ScrambleDB: Oblivious (Chameleon) Pseudonymization-as-a-Service. *PoPETs* 2019, 3 (July 2019), 289–309. https://doi.org/10.2478/popets-2019-0048
[35] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. 1999. Pseudonym Systems. In *SAC 1999 (LNCS, Vol. 1758)*, Howard M. Heys and Carlisle M. Adams (Eds.). Springer, Heidelberg, Germany, Kingston, Ontario, Canada, 184–199. https://doi.org/10.1007/3-540-46513-8_14
[36] Moni Naor, Benny Pinkas, and Omer Reingold. 1999. Distributed Pseudo-random Functions and KDCs. In *EUROCRYPT'99 (LNCS, Vol. 1592)*, Jacques Stern (Ed.). Springer, Heidelberg, Germany, Prague, Czech Republic, 327–346. https://doi.org/10.1007/3-540-48910-X_23
[37] NIST. 2023. *SP 800-63 Digital Identity Guidelines.* https://pages.nist.gov/800-63-4/sp800-63c.html
[38] Christian Paquin and Greg Zaverucha. 2013. U-prove cryptographic specification v1. 1. *Technical Report, Microsoft Corporation* (2013).
[39] Torben P. Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91 (LNCS, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, Heidelberg, Germany, Santa Barbara, CA, USA, 129–140. https://doi.org/10.1007/3-540-46766-1_9
[40] Birgit Pfitzmann and Michael Waidner. 2002. Privacy in browser-based attribute exchange. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society* (Washington, DC) *(WPES '02)*. Association for Computing Machinery, New York, NY, USA, 52–62. https://doi.org/10.1145/644527.644533
[41] David Pointcheval and Olivier Sanders. 2016. Short Randomizable Signatures. In *CT-RSA 2016 (LNCS, Vol. 9610)*, Kazue Sako (Ed.). Springer, Heidelberg, Germany, San Francisco, CA, USA, 111–126. https://doi.org/10.1007/978-3-319-29485-8_7
[42] Ethereum Research. 2023. *Benchmarking ZKP Development Frameworks: the Pantheon of ZKP.* https://ethresear.ch/t/benchmarking-zkp-development-frameworks-the-pantheon-of-zkp/14943

[43] N. Sakimura, J. Bradley, M. Jones, B. de Medeirosk, and C. Mortimore. 2014. *OpenID Connect Core 1.0.* https://openid.net/specs/openid-connect-core-1_0.html
[44] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. 2007. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA) *(CCS '07)*. Association for Computing Machinery, New York, NY, USA, 72–81. https://doi.org/10.1145/1315245.1315256
[45] Rongwu Xu, Sen Yang, Fan Zhang, and Zhixuan Fang. 2023. MISO: Legacy-compatible Privacy-preserving Single Sign-on using Trusted Execution Environments. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 352–372.
[46] Zhiyi Zhang, Michal Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière. 2021. EL PASSO: Efficient and Lightweight Privacy-preserving Single Sign On. *PoPETs* 2021, 2 (April 2021), 70–87. https://doi.org/10.2478/popets-2021-0018

A OMITTED MODEL PARTS

Here, we describe our protocol flow in detail, provide the omitted parts, further explanations of our security model, and compare it to the Session Binding model from [32].

A.1 Correctness

Recall that we denote with $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{U}$ the RP, session, and user spaces. An OPPID scheme – as defined in Sec. 2.3 – is correct if for all $\lambda \in \mathbb{N}$, setup and RP registrations with $rid \in \mathcal{R}$

$$pp \leftarrow \mathsf{Setup}(1^\lambda), ((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp),$$
$$(cred, \mathcal{M}') \leftarrow \langle \mathsf{Join}_{\mathsf{RP}}(ipk, rid), \mathsf{Reg}_{\mathsf{IdP}}(isk, rid, \mathcal{M})\rangle,$$

all authentication sessions with $sid \in \mathcal{S}$ of user $uid \in \mathcal{U}$ to RP rid

$$(orid, crid) \leftarrow \mathsf{AInit}_{\mathsf{U}}(ipk, rid)$$
$$auth \leftarrow \mathsf{AReq}_{\mathsf{RP}}(ipk, rid, cred, crid, orid, sid)$$
$$\tau \leftarrow \mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid, ctx, sid)$$
$$(\tau_{\mathsf{fin}}, ppid) \leftarrow \mathsf{AFin}_{\mathsf{U}}(ipk, rid, crid, orid, ctx, sid, \tau),$$

result in

$$\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid, ppid, ctx, sid), \tau_{\mathsf{fin}}) = 1.$$

A.2 Protocol Flow

We present the protocol flow and explain how parties interact with each other in detail.

Setup and Registration. Before offering its authentication service, the IdP generates its key pair (isk, ipk) based on the public parameters pp and initializes its member state $\mathcal{M}$. The public key ipk is shared with all entities, and tokens issued by the IdP are validated against this key. RPs can then engage in the registration $\langle \mathsf{Join}_{\mathsf{RP}}, \mathsf{Reg}_{\mathsf{IdP}}\rangle$ with the IdP to obtain their credential $cred$.

Authentication Flow. The user authentication to an RP rid via the IdP with ipk involves the following four steps:

(1) The user executes $\mathsf{AInit}_{\mathsf{U}}$ with rid to initiate the authentication process, obtaining $(orid, crid)$. The user then stores $orid$ and transmits both values to the RP.
(2) The RP runs $\mathsf{AReq}_{\mathsf{RP}}$ to generate $auth$ using $crid$, $orid$, and $cred$ to authenticate as a legitimate RP. To ensure freshness, the RP provides a fresh session identifier sid. The user then forwards $auth$ (via the user) to the IdP.
(3) When the IdP receives a token request from a user uid for session sid and implicit authentication $auth$ for an RP, it executes the algorithm $\mathsf{ARes}_{\mathsf{IdP}}$. This results in either a token τ or $\perp$ if the RP authentication fails. The token is now bound to the implicit rid and explicit uid, sid, along with additional session information such as timestamps, simplified through context ctx. We assume that the IdP has properly authenticated uid, but do not make that explicit here.
(4) The user runs $\mathsf{AFin}_{\mathsf{U}}$ to transform the IdP's token τ with the committed rid to verify that the final token corresponds to the initial rid and to derive an RP-specific pseudonym $ppid$. This algorithm takes the user opening $orid$ and all received information as input to produce the final token τ_{fin} and $ppid$.

The resulting token τ_{fin} is then verified against ipk to confirm its validity for the tuple $(rid, ppid, ctx, sid)$. This explicit verification binds the session information, $ppid$, and the RP's rid together.

A.3 Session Binding: RP Binding & RP Auth.

Here, we provide more intuition on how the properties of RP Binding and RP Authentication are captured in our Session Binding game $\mathsf{Exp}^{\mathsf{SES\text{-}BIN}}_{\mathcal{A},\mathsf{OPPID}}$ in Fig. 3. Note that RP Binding II and RP Authentication II define security aspects that were not covered by the original model in [32]. We give a more detailed comparison in the following section.

RP Binding. If an honest user uid wants to authenticate to a certain RP rid in a session sid, the adversary wins if it can produce a token for the same session, but which is valid for a different rid^*. This models phishing attacks from malicious RPs. To capture this property, we need to know what the intent of the honest user was, which is done through $\mathcal{O}.\mathsf{AInit}_{\mathsf{U}}$ and $\mathcal{O}.\mathsf{AResFin}$. Using the bookkeeping in these oracles, we can define an honest user's intent as (rid, uid, ctx, sid) in $\mathsf{Q}_{\tau_{\mathsf{fin}}}$ when the finalized token is computed. The adversary wins if it can come up with a token for that session but with $rid^* \neq rid$, as defined in condition (b) for *RP Binding I*.

For a session initiated by a corrupt user towards a corrupt RP, we never know the exact RP the user wants to authenticate to: $\mathcal{A}$ invokes $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ on adversarial chosen inputs $auth$ and $crid$, which both hide the rid. So our guarantees are naturally weaker here than for honest users. What we do guarantee, is that the adversary cannot re-use the same token it receives from the IdP, and claim it to be valid towards multiple corrupt RPs. This is the reason we have $\mathcal{O}.\mathsf{Vf}_{\mathsf{RP}}$. It might be surprising that we provide an oracle that runs purely on public values, and would mimic something that is typically run internally be the adversary. What we want to model here is that if the adversary presents a token for $(rid, ppid, ctx, sid)$ somehow *publicly*, e.g., towards a judge or some external *honest* entity, and later presents a token for the same context ctx, sid but for a different $rid^* \neq rid$ (which is its forgery). Thus, despite the honest IdP never learning the exact RP the corrupt user wanted to authenticate, we guarantee that this user can authenticate to at most *one* RP. This *RP Binding II* is captured in winning condition (c), and can be seen as a notion similar to the one-more unforgeability property in blind signatures.

RP Authentication. The previous two properties ensured that the IdP-generated token is bound to the blindly received rid. We further want to guarantee that only legitimate RPs can request such tokens, which is captured in conditions (d) and (e). In condition (d), the adversary wins if it manages to produce a valid token for some rid^*, yet this RP has never registered, i.e., $rid^* \notin \mathsf{HRP} \cup \mathsf{CRP}$ where HRP and CRP are the sets of all honest and corrupt RPs, the IdP has registered through $\mathcal{O}.\mathsf{RegHRP}$ and $\mathcal{O}.\mathsf{RegCRP}$.

If the rid^* in the forgery belongs to an honest RP, we want even stronger guarantees and ensure that it must be infeasible to win if the honest RP had never authenticated for that particular session sid^*. As all authentication requests for honest RPs are handled through $\mathcal{O}.\mathsf{AReq}_{\mathsf{RP}}$, where we store each query (rid, sid) in $\mathsf{Q}_{\mathsf{auth}}$, this translates to simply checking that $(rid^*, sid^*) \notin \mathsf{Q}_{\mathsf{auth}}$.

A.4 Comparison to Session Binding from [32]

Before we compare the security guarantees of our Session Binding model to the one from [32], we outline two fundamental *functional* differences between both models.

Functional Differences. The focus of our work is *pseudonymous* user authentication towards RPs, where the pseudonyms are blindly computed by the IdP. This was the main challenge and manifests as Nym Correctness in our model. In the work of Kroschewski and Lehmann [32], all authentication tokens contained the globally unique user identifier *uid* that was directly vouched for by the IdP.

The focus of [32] was on RP authentication, and their work explicitly models epoch-based credentials to allow adaptive RP corruption and enable their revocation. Our model omits this approach for simplicity. We stress that epoch-based credentials are straightforward to add at the construction level but significantly complicate the security model. In fact, epoch-based renewal is an orthogonal aspect to our focus on pseudonymous identifiers, and we can consider our setting as a single-epoch – and pseudonym-extended – version of [32].

Original Model Only Considers Honest Users + Corrupt RPs. For better comparison, we state the original Session Binding property translated to our single-epoch view and syntax on the left in Fig. 5. This makes it easy to see that, ignoring the obvious differences due to the different functional properties, our Session Binding model is significantly stronger than [32], as shown on the right in Fig. 5.

The original definition has the weakness that it only considers the security of sessions between honest users and corrupt RPs. This was justified by the argument that (i) honest users are necessary in order to know the *rid* a user wanted to authenticate to in a particular session and compare it to the adversary's forgery, and (ii) honest RPs do not give the adversary an advantage. We will now explain why this restriction to honest users and corrupt RPs led to a security model that does not capture all desirable properties, and how we incorporated them into our definition.

Adding Security for Sessions of Corrupt User. Regarding (i): While we indeed do not know the intended *rid* when an IdP issues a token towards a corrupt user and corrupt RP, the model should not abandon security in such scenarios. What we still care about -— and in fact, this could be seen as the most crucial security property of SSO -— is that the IdP's signature cannot be used out of context. Simply removing the restriction of honest users in the winning condition already enhances security. This is what our game achieves compared to [32] for Direct Forgeries (1) and Indirect Forgeries (2a, d). However, this alone is not sufficient: the guarantee does not extend to the blindly signed *rid*, as condition (2b) is the only subcase that strictly needs to be limited to the honest user setting.

This is where our game introduces the winning condition (2c) in a one-more unforgeability style. This condition requires that it must be infeasible to produce multiple valid tokens for different corrupt *rid*s. Generating multiple tokens for corrupt users and RPs would not be a direct attack on the authentication session, but it would still be undesirable behavior: it could allow corrupt RPs to present apparently IdP-attested tokens for non-existent sessions, falsely inflating their active user base. Our stronger model prevents this.

Adding Stronger Guarantees for Honest RPs. The second enhancement is related to (ii). The original model allows the adversary to win by producing a token for an *rid* that was never registered at all. Again, this was only defined for honest users, whereas our model extends this to corrupt users as well. The more significant change is that our model provides stronger guarantees when an *honest* RP is involved—which was not addressed in [32]. Our winning condition (2e) additionally requires that it must be infeasible to produce a token for some *rid*, *sid* when *rid* belongs to an honest RP that never authenticated for session *sid*.

At first glance, this might appear to be covered by the RP Accountability game in [32]. However, the RP Accountability definition only applies to fully blind authentication requests – and therefore is weaker. In our Session Binding game, we have knowledge of the finalized token including *rid*, and can verify whether a corresponding authentication request from this *rid* and session *sid* exists.

A.5 Request Authentication

This property captures the authenticity of the *request* that an IdP receives and uses to produce its token. It ensures that if an IdP receives an authenticated request $(auth, crid, sid)$ via a user *uid* for which $\mathsf{ARes}_{\mathsf{IdP}}(isk, auth, crid, uid, ctx, sid)$ produces an output $\neq \perp$, it must originate from a previously registered RP.

The game is defined by $\mathsf{Exp}^{\mathsf{REQ\text{-}AUTH}}_{\mathcal{A},\mathsf{OPPID}}$ (see Fig. 3) and follows a classic unforgeability definition. Instead of demanding security for the final authentication token (as in Session Binding), it captures unforgeability for the RP authenticated information: *crid*, *sid*. The IdP is honest here, and the adversary can register honest RPs through $\mathcal{O}.\mathsf{RegHRP}$, obtain their authenticated requests through $\mathcal{O}.\mathsf{AReq}_{\mathsf{RP}}$, and query tokens from the IdP through $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$. The adversary wins if it outputs $(auth^*, crid^*, uid^*, ctx^*, sid^*)$, where the IdP accepts the authentication (i.e., does not return $\perp$) and no honest RP authenticated $crid^*$ in session sid^*.

Note that this property defines the security of the fully blind authentication request towards the IdP, which implies that all RPs must be honest -— otherwise, "forging" is trivial. The original definition in [32] allowed corrupt RPs, which was possible as their work considered epoch-based renewal of membership credentials. The security model then only requires that all registered RPs of the *current* epoch must be honest, but RPs from previous epochs can be corrupt. In this sense, just as in our Session Binding definition, our Request Authentication definition can be seen as a single-epoch version of the Request Authentication definition of [32]. We do not

$\mathsf{Exp}^{\text{RP Session Binding}}_{\mathcal{A},\text{OPPID}}(\lambda)$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$; $((isk, \mathcal{M}), ipk) \leftarrow \mathsf{KGen}_{\mathsf{IdP}}(pp)$
$O := \{\mathsf{RegCRP}, \mathsf{AInit}_{\mathsf{U}}, \mathsf{AReq}_{\mathsf{RP}}, \mathsf{ARes}_{\mathsf{IdP}}, \mathsf{AResFin}\}$
$(rid^*, ppid^*, ctx^*, sid^*, \tau^*_{\mathsf{fin}}) \leftarrow \mathcal{A}^O(ipk)$
Return 1 if $\mathsf{Vf}_{\mathsf{RP}}(ipk, (rid^*, uid^*, ctx^*, sid^*), \tau^*_{\mathsf{fin}}) = 1 \wedge$ "uid^* is honest"
and at least one of the following holds:
(a) $(rid^*, uid^*, ctx^*, sid^*) \notin \mathsf{Q}_{\tau_{\mathsf{fin}}}$
(b) $(rid^*, uid^*, ctx^*, sid^*) \in \mathsf{Q}_{\tau_{\mathsf{fin}}} \wedge rid^* \notin \mathsf{CRP}$

This Work	Model [32]
(1) Direct Forgery	(a), but for honest users only
(2) Indirect Forgery:	
(a) Nym Correctness	(a), via *uid* correctness, but for honest users only
(b) RP Binding I	(a)
(c) RP Binding II	–, missing
(d) RP Authentication I	(b), but for honest users only
(e) RP Authentication II	–, missing, partially covered via RP Accountability

Figure 5: Left: RP Session Binding from [32] translated to our syntax, where we abuse notation and consider $\mathsf{F}(uid, rid) = uid$. The honest user requirement was enforced in [32] by checking that the uid^* of the forgery was never used in a "malicious" query via $O.\mathsf{ARes}_{\mathsf{IdP}}$ but always via $O.\mathsf{AResFin}$ – which allows the challenger to know the intended rid for every session. Right: Comparison between the Session Binding guarantees in this work and [32], mapping the winning conditions to each other.

allow any corrupt RPs in the definition as any corrupt RP in the single-epoch would allow the adversary to win trivially.

Definition A.1 (Request Authentication). An OPPID scheme satisfies Request Authentication if for all PPT adversaries $\mathcal{A}$, it holds

$$\Pr[\mathsf{Exp}^{\text{REQ-AUTH}}_{\mathcal{A},\text{OPPID}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

B BUILDING BLOCKS

This section introduces the necessary building blocks. The security parameter is denoted as $\lambda \in \mathbb{N}$, and the symbol $\perp$ represents failure. Note that all algorithms may use global parameters pp, such as shared groups, instead of 1^τ, and may also provide additional public parameters. For simplicity, we omit explicit mention of these public parameters or the algorithms used to generate them.

Commitment Scheme. A commitment scheme COM = (Com, Open) produces a commitment *com* and its corresponding opening o using the algorithm $\mathsf{Com}(m)$. The algorithm $\mathsf{Open}(m, com, o)$ outputs 1 if *com* is a valid commitment for m, and 0 otherwise. The commitment scheme must satisfy hiding and binding properties.

Non-interactive Zero-Knowledge Proofs. In a *non-interactive* zero-knowledge proof system [6, 22], the prover and verifier possess the statement s and some public context x. The prover generates a proof $\pi \leftarrow \mathsf{NIZK}\{(w) : s(w)\}(x)$ that convinces the verifier that $s(w) = 1$, without revealing w to the verifier and ensuring that π is bound to x. We require the proof system to be zero-knowledge and simulation-sound [25].

Signature Scheme. A signature scheme is a tuple of algorithms $\mathsf{S}_1 = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, with key generation $(pk, sk) \leftarrow \mathsf{KGen}(1^\lambda)$, signing $\sigma \leftarrow \mathsf{Sign}(sk, m)$, and verification as $0/1 \leftarrow \mathsf{Vf}(pk, m, \sigma)$. We need S_1 to be Existentially Unforgeable under a Chosen Message Attack (EUF-CMA) [24]. In our implementation, we use RSA signatures for compatibility with existing standards.

We also require a signature scheme $\mathsf{S}_2 = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ that supports the creation of efficient NIZKs. The NIZK should prove knowledge of a valid signature σ on a message m under pk without revealing the message or signature. In our construction, we combine S_2 signatures with commitments using a NIZK proof that demonstrates knowledge of a signature on a committed message as

$$\mathsf{NIZK}\{(m, \sigma, o) : \mathsf{Vf}(pk, m, \sigma) = 1 \wedge \mathsf{Open}(m, com, o) = 1\}(com)$$

This proof discloses only the commitment *com* while verifying the possession of a valid signature σ under pk on m and an opening o to the commitment *com* for the signed message. We instantiate this scheme with PS signatures [41], which provide all these features.

Pseudorandom Functions. We require a pseudorandom function $y \leftarrow \mathsf{PRF}(k, x)$ that produces output indistinguishable from random, towards an adversary not knowing the key k. We need two different pseudorandom functions, one that produces pseudorandom values in $\mathbb{Z}_q$ and can simply be HMAC with a proper output mapping; and a second function that allows for the partially-blind evaluation needed in our protocol. For the latter, we use the DL-based $\mathsf{PRF}(k, x) := \mathsf{H}(x)^k$ [36] in a group $\mathbb{G}$ of prime order q.

C FULL PROOFS OF π_{OPPID}

Here, we provide the proofs showing that our protocol described in Sec. 4 achieves all properties defined in Sec. 3. We start by analyzing the properties of the pseudonym function $\mathsf{F}_{\mathsf{DL+PRF}}$.

Note that the simple proofs for Unobservability and Request Authentication are omitted, as well as the full reductions, they are all available in the full version though.

Properties of $\mathsf{F}_{\mathsf{DL+PRF}}$. We analyze the properties of our pseudonym function

$$\mathsf{F}_{\mathsf{DL+PRF}} = \mathsf{F}_{\mathsf{k}}(uid, rid) := \mathsf{H}(rid)^{\mathsf{PRF}(k, uid)}.$$

Recall that our OPPID model requires this function to provide unique, collision-free, and unlinkable pseudonyms (see Sec. 2.2).

Uniqueness requires that for every rid, uid combination, there is a unique pseudonym $ppid = \mathsf{F}(uid, rid)$. $\mathsf{F}_{\mathsf{DL+PRF}}$ naturally ensures uniqueness as it is a deterministic function.

Regarding collision-freeness, note that $\mathsf{H}(rid)^{uk}$ is a permutation for $uk \in \mathbb{Z}_q$. If $uk \leftarrow \mathsf{PRF}(k, uid)$ is an injective function, then F provides different pseudonyms for each user. That is, for all rid and $uid \neq uid'$ it holds that $\mathsf{F}_{\mathsf{k}}(uid, rid) \neq \mathsf{F}_{\mathsf{k}}(uid', rid)$. This property requires that $\mathcal{U} \leq \mathbb{Z}_q$, which holds for any normal deployment, where the number of users is significantly smaller than $\mathbb{Z}_q$.

Lemma C.1. *Our pseudonym function* $\mathsf{F}_{\mathsf{DL+PRF}}$ *provides unique and collision-free pseudonyms, if* H *is deterministic,* PRF *is deterministic, injective, and* $\mathcal{U} \leq \mathbb{Z}_q$.

In the following, we prove that $\mathsf{F}_{\mathsf{DL+PRF}}$ is a secure pseudorandom function. This immediately grants the unlinkability of the pseudonyms, which will be helpful in proving the Unlinkability property of π_{OPPID}. Note that a similar function (essentially $\mathsf{F}_{\mathsf{DL+PRF}}$ with double hashing) was shown to be a secure (partially oblivious) pseudorandom function already [31], whereas we need the classic PRF property here and do not apply the outer hash.

Lemma C.2. *Our pseudonym function* $\mathsf{F}_{\mathsf{DL+PRF}}$ *is a secure PRF if* H *is a random oracle,* PRF *is a secure pseudorandom function, and DDH holds in* $\mathbb{G}$.

Proof. Our proof has two main steps. The first step switches from $\mathsf{PRF}(k, uid)$ to the random values from $uk \leftarrow_{\mathsf{R}} \mathbb{Z}_q$ while evaluating the PRF output in the oracle. This change is indistinguishable by pseudorandomness of PRF. The second step relies on the simple observation that $\mathsf{H}(rid)^{uk}$ is HashDH PRF [36] and shows the pseudorandomness of these values by relying on the multi-key pseudorandomness of the HashDH PRF.

Game 0. This game is identical to the pseudorandomness game with $\mathsf{F}_{\mathsf{DL+PRF}}$.

Game 1. In this game, for each PRF query (uid_i, rid) with a new uid_i, we sample $uk_i \leftarrow_{\mathsf{R}} \mathbb{Z}_q$ for the uid_i and answer $\mathsf{F}_{\mathsf{DL+PRF}}(uid_i, rid)$ queries as $\mathsf{F}_{\mathsf{DL+PRF}}(uid_i, rid) = \mathsf{H}(rid)^{uk_i}$. By pseudorandomness of PRF, this change is indistinguishable to the adversary.

Game 2. This game finalizes the proof by changing the PRF evaluations for $\mathsf{F}_{\mathsf{DL+PRF}}(uid_i, rid)$ from $\mathsf{H}(rid)^{uk_i}$ to $y \leftarrow_{\mathsf{R}} \mathbb{G}$.

We show that this change is indistinguishable to the adversary by presenting sequences of indistinguishable hybrids between Games 1 and 2 where the first and last hybrids are identical to Games 1 and 2, respectively. Each hybrid changes the PRF evaluations for uid_i from $\mathsf{H}(rid)^{uk_i}$ to a random value. Let $uid_1, ..., uid_n$ represent the uid values that the adversary queries $\mathsf{F}_{\mathsf{DL+PRF}}$ oracle.

Hybrid$_0$. This hybrid is identical to Game 1.

Hybrid$_{i\in\{1,...,n\}}$. Hybrid$_i$ runs Hybrid$_{i-1}$ identically except for the following change. For $\mathsf{F}_{\mathsf{DL+PRF}}(uid_i, rid)$ queries, Hybrid$_i$ outputs $\mathsf{F}_{\mathsf{DL+PRF}}(uid_i, rid) \leftarrow_{\mathsf{R}} \mathbb{G}$ random values. Note that Hybrid n is identical to Game 2.

Transition Hybrid$_i$ $\rightarrow$ *Hybrid*$_{i+1}$. The transition between hybrids simply relies on the pseudorandomness of HashDH. In particular, Hybrid$_i$ evaluates HashDH PRF $\mathsf{H}(rid)^{uk_{i+1}}$ with the PRF key uk_{i+1} whereas Hybrid$_{i+1}$ changes these evaluations to the random values from $\mathbb{G}$. By pseudorandomness of HashDH, Hybrids i and $i+1$ are indistinguishable and HashDH is pseudorandom if DDH assumption holds in ROM [36]. □

C.1 Proof of Theorem 4.1 (Unlinkability)

Here, we prove that π_{OPPID} satisfies Unlinkability (see Def. 3.1) if H is a random oracle, PRF is a secure pseudorandom function, and the DDH assumption holds in $\mathbb{G}$.

Proof. Recall that $\mathcal{A}$ is given the IdP's public key and oracles $\mathcal{O} := \{\mathsf{RegCRP}, \mathsf{AInit}_{\mathsf{U}}, \mathsf{ARes}_{\mathsf{IdP}}\}$ to register corrupt RPs, initialize honest user sessions, and obtain pseudonyms and tokens from the IdP. Eventually, it outputs uid_0, uid_1, and rid, together with $auth, crid, ctx, sid$, and receives a challenge token and pseudonym $(\tau_{fin_b}, ppid_d)$ for uid_b.

We further know that if $\mathcal{A}$ wins, it must not make any query that trivially reveals $\mathsf{F}_{\mathsf{k}}(uid_0, rid)$ or $\mathsf{F}_{\mathsf{k}}(uid_1, rid)$. That is, $\mathcal{A}$ is not allowed to make $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ queries $(auth, crid, uid_d, ctx, sid)$ for $d \in \{0, 1\}$ where either: $crid$ belongs to rid from the challenge query (via previous query to $\mathcal{O}.\mathsf{AInit}_{\mathsf{U}}$), or $crid$ is malicious, i.e., *not* an output from $\mathcal{O}.\mathsf{AInit}_{\mathsf{U}}$.

These conditions ensure that all queries $\mathcal{A}$ makes towards the oracle $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ must involve $crid$ values that are honestly generated and for which we know the underlying rid (and blinding value r).

This makes the proof straightforward. As we already know the blinding value of $\overline{y}$, we can compute the response value by relying on PRF computation instead of an oblivious PRF evaluation. We prove Unlinkability through a small sequence of games, replacing all user-dependent values for uid_0 and uid_1 with random values, ensuring that $\mathcal{A}$ has no better chance than guessing to determine the bit b. Let *Game 0* be the original game.

Game 1. We now change the way we compute the $\overline{y}_d$ values in $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ to compute the challenge token values τ_d. We simulate $\mathcal{A}$'s view by using $\mathsf{F}_{\mathsf{DL+PRF}}$ as a black-box algorithm to reason about its pseudorandomness, showing the Unlinkability property of our scheme instead of its one-more pseudorandomness. Since $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ queries must contain $crid$ values generated through $\mathsf{AInit}_{\mathsf{U}}$, we can look up $(rid, crid, orid)$ in $\mathsf{Q}_{\mathsf{rid}}$ and parse $orid = (o, r)$. Instead of blindly computing $\overline{y}$, we compute $y = \mathsf{F}_{\mathsf{DL+PRF}}$ $(k, (rid, uid))$ from the known input and blind the response later: $\overline{y}_d = y_d^r$. This change produces outputs identical to those in the previous game, so this game hop cannot be distinguished by $\mathcal{A}$.

Game 2. In this game, we change the way we compute $y_d = \mathsf{F}_{\mathsf{DL+PRF}}(k, (uid_d, rid))$ while computing τ_d values and set $y_d \leftarrow_{\mathsf{R}} \mathbb{G}$ as a random value. By the winning condition of the Unlinkability game, we know that there are no previous $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ queries for (uid_d, rid). Also, by the previously proven Lemma C.2, we know that $\mathsf{F}_{\mathsf{DL+PRF}}$ is a secure PRF, so the change in this game is indistinguishable.

In the last game, all the bit d-related values are uniformly random, independent of uid_0 and uid_1. Thus, $\mathcal{A}$'s chance of determining the bit b is 1/2. □

No One-More-Type Assumption. What might be surprising at first glance is that we do not need a one-more-type assumption typically required in blind evaluation protocols, such as the (2)HashDH-OPRF, which seems equivalent to our construction. This is not surprising after closer examination. First, recall that our security property guarantees that the *uid* in the IdP's response remains hidden, and the *uid* is *not* blind towards the IdP but revealed in every query. Second, the guarantee can only hold for *honest* users, which is enforced throughout the game for both challenge users, uid_0 and uid_1. The adversary can only obtain "blind" PRF evaluations (as part of $\mathcal{O}.\mathsf{ARes}_{\mathsf{IdP}}$ queries), where the blinded input $\overline{x}$

was honestly generated. Thus, in the security proof, the challenger is always aware of the blinded rid behind $\overline{x}$, i.e., knows exactly on which values the PRF $\mathsf{F}_{\mathsf{DL+PRF}}$ is evaluated. This allows us to prove Unlinkability under the standard assumption that $\mathsf{F}_{\mathsf{DL+PRF}}$ is a secure pseudorandom function, which holds if H is a random oracle, and the DDH assumption holds in $\mathbb{G}$ [31].

C.2 Proof of Theorem 4.2 (Unobservability)

We now provide a simple proof that π_{OPPID} satisfies Unobservability (see Def. 3.2) if COM is hiding, and the NIZK is zero-knowledge.

Proof. $\mathcal{A}$ receives $auth = \pi$ and $crid = (\overline{x}, com)$, where com is a commitment to rid and $\overline{x}$ is the blinded hash. π proves knowledge of an IdP-issued credential on rid and that the commitment com opens to the same rid. Unobservability follows directly from the zero-knowledge property of π, the perfect hiding of rid via $\overline{x}$ with r, and the hiding property of COM with the undisclosed opening o. □

C.3 Proof of Theorem 4.3 (Session Binding)

Here we prove that π_{OPPID} is Session Binding (see Def. 3.3) if the S_1 and S_2 schemes are both EUF-CMA secure, COM is binding, and the NIZK is zero-knowledge and simulation-extractable.

Proof. We split the proof along the winning condition that the adversary must satisfy. Recall that $\mathcal{A}$ outputs $(rid^*, ppid^*, ctx^*, sid^*, \tau^*_{\mathsf{fin}})$ and wins if this is a valid yet non-trivial forgery.

The final token τ^*_{fin} contains $(com^*, o^*, r^*, \overline{y}^*, \sigma^*_\tau)$. A valid token implies that σ^*_τ is a valid signature, meaning

$$\mathsf{S}_1.\mathsf{Vf}(pk_1, (com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*), \sigma^*_\tau) = 1$$

with $\overline{x}^* := \mathsf{H}(rid^*)^{r^*}$, $com^* := \mathsf{Com}(rid^*, o^*)$, and $ppid^* := \overline{y}^{*-r^*}$.

We first distinguish whether we have a *direct* or *indirect forgery*, i.e., whether the information that was publicly signed by the IdP is already a forgery or not.

Case 1: Direct Forgery with $(\cdot, ctx^*, sid^*) \notin \mathsf{Q}_\tau$. If the adversary outputs a valid forgery where the combination $(\cdot, ctx^*, sid^*)$ was never vouched for by the honest IdP, it must have forged the IdP's signature on these values. This is infeasible if the signature scheme S_1 is EUF-CMA secure.

We can build a S_1 forger easily using an adversary that can perform a direct forgery as follows. We get the challenge S_1 public key pk^*_1 and use it as pk_1 while setting the issuer public key ipk. As we do not know the corresponding secret key to pk^*_1, we need a way to simulate the S_1 signatures σ_τ which are part of IdP's output to τ queries, τ. We simulate σ_τ values using the signing oracle of S_1. In particular, the oracles work as follows:

$\mathsf{RegHRP}, \mathsf{RegCRP}, \mathsf{AInit}_{\mathsf{U}}, \mathsf{AReq}_{\mathsf{RP}}, \mathsf{AResFin}, \mathsf{Vf}_{\mathsf{RP}}$: As they are.

$\mathsf{ARes}_{\mathsf{IdP}}$: It computes $\overline{y}$ as before. To form the token on $m = (com||\overline{x}||\overline{y}||ctx||sid)$, it queries the signing oracle of S_1 and gets the signature σ_τ on m and outputs $(\sigma_\tau, \overline{y})$.

Finally, the adversary outputs $(\sigma^*, m^*) := (\sigma^*_\tau, (com^*, \overline{x}^*, \overline{y}^*, ctx^*, sid^*))$ to the S_1 unforgeability challenger as the S_1 forgery for the forged token $\tau^*_{\mathsf{fin}} := ((com^*, o^*, r^*, \overline{y}^*, \sigma^*_\tau), y^*)$ and $\overline{x}^* := \mathsf{H}(rid^*)^{r^*}$. As the forged token is valid, we know that $\mathsf{S}_1.\mathsf{Vf}(pk^*_1, \sigma^*, m^*) = 1$. Furthermore, direct token forgeries ensure that we do not make a signing oracle query for $(\cdot||\cdot||\cdot||ctx^*||sid^*)$, so our forgery is on a fresh message.

Case 2: Indirect Forgery with $(uid, ctx^*, sid^*) \in \mathsf{Q}_\tau$. If the honest IdP has previously signed the combination (uid, ctx^*, sid^*) for a session for user uid, the adversary can only win if the associated information $(rid^*, ppid^*)$ that the IdP has blindly signed and derived is inconsistent with the expected pseudonym or behavior of honest and corrupt RPs. This inconsistency is expressed through the five sub-cases in the winning condition of the Session Binding game, and our proof branches accordingly.

What is important here is that the IdP signs additional information in σ^*_τ, namely $(com^*||\overline{x}^*||\overline{y}^*)$. If the adversary outputs a forgery where this tuple differs from what the honest IdP has signed along with $(ctx^*||sid^*)$, then we can immediately turn this into a forgery of the S_1 scheme. We simulate pk_1 as the S_1 EUF-CMA challenge public key pk^*_1 and simulate the Session Binding game just as in Case 1.

Note that while the adversary does not explicitly output com^* and $\overline{x}^*$, these values are uniquely defined through its outputs as $com^* = \mathsf{Com}(rid^*, o^*)$ and $\overline{x}^* = \mathsf{H}(rid^*)^{r^*}$, with o^* and r^* being part of τ^*_{fin} and rid^*. At the end of the Session Binding game, we check the $\mathsf{ARes}_{\mathsf{IdP}}$ query with $(uid, ctx^*, sid^*) \in \mathsf{Q}_\tau$. If the corresponding query differs from $(com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*)$, then we can output $(\sigma^*_\tau, (com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*))$ as a valid S_1 forgery.

Thus, the rest of the proof of Case 2 is now conditioned on the fact that the full tuple $(com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*)$ has indeed been signed by the honest IdP in a session with user uid.

(a) *Nym Correctness*: $ppid^* \neq \mathsf{F}(uid, rid^*)$. If the adversary wins by satisfying the first sub-condition, it must have produced a valid token with an incorrect pseudonym, i.e., output a $ppid^* \neq \mathsf{H}(rid^*)^{\mathsf{PRF}(k, uid)}$.

In our construction, winning under this condition is impossible (other than through manipulating the IdP's signed information, which we excluded above). Recall that the adversary's forgery must contain $\overline{x}^*||\overline{y}^*$ along with the public session information. We have already excluded the case where $\mathcal{A}$ manages to manipulate these values. Thus, we know that $\overline{x}^*, \overline{y}^*$ are the values the IdP has signed in a session sid^*, where it learned the username uid and computed

$$\overline{y}^* = \overline{x}^{*\mathsf{PRF}(k, uid)}.$$

As the forgery must pass the verification, we know that

$$ppid^* = \overline{y}^{*-r^*} \quad \text{and} \quad \overline{x}^* = \mathsf{H}(rid^*)^{r^*}$$

Putting it all together implies that

$$ppid^* = ((\mathsf{H}(rid^*)^{r^*})^{\mathsf{PRF}(k, uid)})^{-r^*} = \mathsf{H}(rid^*)^{\mathsf{PRF}(k, uid)}.$$

Thus, for every valid token, it holds that $ppid^* = \mathsf{F}(uid, rid^*)$.

(b) *RP Binding I*: $(rid, uid, ctx^*, sid^*) \in \mathsf{Q}_{\tau_{\mathsf{fin}}} \land rid \neq rid^*$. If an adversary wins under condition (2b), it must have "high-jacked" an honest user session. For sessions intended by honest users, we know the exact RP rid they intended to authenticate to, and the adversary wins if it can create a token for this session that is valid for a different RP $rid^* \neq rid$. We can again leverage the fact that we know that $(com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*)$ is the original information signed by the honest IdP in the session with the honest user uid. We further know that the commitment com^* is an honestly generated commitment (through $\mathcal{O}.\mathsf{AInit_U}$) for rid.

As the final token contains an opening o^* and checks that $com^* = \mathsf{Com}(rid^*, o^*)$, this implies that $\mathcal{A}$ managed to open the commitment com^* to a different value $rid^* \neq rid$, which is infeasible under the binding property of COM. In particular, by the behavior of $\mathcal{O}.\mathsf{AResFin}$, we know that $\mathsf{Q}_{\tau_{\mathsf{fin}}}$ is updated with an rid only when there is a valid opening o for the commitment o^* provided in the token τ_{fin}. Thus, the tuple $(com^*, rid, rid^*, o, o^*)$ breaks the binding property of the commitment.

(c) *RP Binding II*: $(rid, ppid, ctx^*, sid^*) \in \mathsf{Q_{vf}} \wedge rid \neq rid^*$. If the IdP did sign $(com^*||\overline{x}^*||\overline{y}^*||ctx^*||sid^*)$ in a session with a *corrupt* user uid (which was ensured by the main $\mathsf{S_1}$ unforgeability reduction of condition (2)), we do not know the intended rid contained in com^* (or $\overline{x}^*$). Thus, the adversary can "open" the token to any valid rid^* it wants. In order to create a valid forgery, $\mathcal{A}$ must have produced (at least) *two* valid tokens for (ctx^*, sid^*) yet different $(rid, ppid)$ and $(rid^*, ppid^*)$ with $rid \neq rid^*$. Note that we do not make any requirements on the pseudonyms here, as the adversary already wins under condition (2a) if it can produce an invalid pseudonym.

The only remaining way for the adversary to provide a condition (2c) forgery is then providing different openings of com^* to distinct rid and rid^* values. If this occurs, the query of the adversary to $\mathcal{O}.\mathsf{Vf_{RP}}$ with sid^* and ctx^* contains a valid opening of com^* to rid and o for $rid^* \neq rid$. Obviously, it contradicts the binding property of COM as the tuple $(com^*, rid, rid^*, o, o^*)$ breaks the binding property.

(d) *RP Authentication I*: $rid^* \notin \mathsf{HRP} \cup \mathsf{CRP}$. If the adversary wins under this condition, it means that the adversary produced a valid token for an rid^* that was never registered with the honest IdP. As the IdP only provides an authentication token when it receives a valid registration proof π, $\mathcal{A}$ must have forged this proof in its query for sid^*. $\mathcal{A}$ can perform such an attack either by forging a proof on an invalid statement directly, or by forging the underlying witness, which is a tuple in the form of (rid, o, σ_{rid}). Here, by forging a witness, we mean either forging the membership credential σ_{rid} on a non-registered rid, or forging the opening (rid^*, o^*) of com^* where (rid, o) is also a valid opening to com^* for a registered but corrupted rid. Neither of these cases is feasible by relying on the special soundness of NIZK, EUF-CMA of $\mathsf{S_2}$ under pk_2, and the binding property of the commitment scheme. To be able to reduce to a forgery under $\mathsf{S_2}$, we require NIZK to be special sound and use the knowledge extractor to obtain a valid witness (rid, o, σ_{rid}). By relying on the binding property of the commitment scheme, we can argue that σ_{rid} satisfies the winning condition of the EUF-CMA game.

In more detail, our reduction in condition (2d) works as follows. We obtain a challenge public key pk_2^* from a $\mathsf{S_2}$ unforgeability challenger and simulate the pk_2 in the identity provider public key ipk as pk_2^*. We simulate the $\mathsf{S_2}$ signatures for $\mathcal{O}.\mathsf{RegCRP}/\mathcal{O}.\mathsf{RegHRP}$ queries by relying on the signing oracle of the $\mathsf{S_2}$ unforgeability challenger, so the behavior of the oracles changes as follows:

$\underline{\mathsf{AInit_U}, \mathsf{AReq_{RP}}, \mathsf{ARes_{IdP}}, \mathsf{AResFin}, \mathsf{Vf_{RP}}}$**:** As they are.

$\underline{\mathsf{RegHRP}}$**:** It checks that $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$. If it does not hold, it outputs 0. Otherwise, for the registration query for rid, it queries the $\mathsf{S_2}$ signing oracle, gets the signature σ_{rid}, updates $\mathsf{HRP} := \mathsf{HRP} \cup \{(rid, \sigma_{rid})\}$ on rid, and outputs 1.

$\underline{\mathsf{RegCRP}}$**:** It checks that $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$. If it does not hold, it outputs 0. Otherwise, for the registration query for rid, it queries the $\mathsf{S_2}$ signing oracle, gets the signature σ_{rid}, updates $\mathsf{CRP} := \mathsf{CRP} \cup \{(rid, \sigma_{rid})\}$ on rid, and outputs σ_{rid}.

Finally, we run the knowledge extractor for the NIZK on the proof π which corresponds to the $\mathcal{O}.\mathsf{ARes_{IdP}}$ query for $(uid, ctx, sid) \in \mathsf{Q}_\tau$ and extract a valid witness (rid, o, σ_{rid}). By the special soundness property of the underlying NIZK, we know that the extractor will output a valid witness with overwhelming probability, so $\mathsf{S_2}.\mathsf{Vf}(pk_2, rid, \sigma_{rid})$ and $\mathsf{Open}(rid, com, o)$.

If $rid \neq rid^*$, then we break the binding property of the commitment scheme as (rid, o) and (rid^*, o^*) are distinct valid openings to the commitment com^*. Otherwise, σ_{rid} is a valid signature on $rid^* = rid$. There is no signing oracle query to the $\mathsf{S_2}$ challenger for rid^* by condition (2d), so (σ_{rid}, rid^*) is a valid and fresh $\mathsf{S_2}$ forgery, breaking the EUF-CMA property of $\mathsf{S_2}$.

We conclude that if there is a type (2d) forger adversary, the underlying NIZK is not special sound, the underlying commitment scheme is not binding, or $\mathsf{S_2}$ is not EUF-CMA.

(e) *RP Authentication II*: $rid^* \in \mathsf{HRP} \wedge (rid^*, sid^*) \notin \mathsf{Q_{auth}}$. When the adversary wins by satisfying the final condition, it has created a valid token for session sid^* and honest RP rid^*, yet this RP never provided authentication for that session. As in the previous case, $\mathcal{A}$ can do this by forging π directly, finding a commitment collision for rid^* and a corrupted rid, or forging the membership credential σ_{rid} of the honest RP. To formally prove it, we use both the zero-knowledge and simulation extractability properties here.

We aim to show that the adversary must forge a NIZK proof π or a witness (rid, o, σ_{rid}) as explained in condition (2d). Thus, similar to condition (2d), we must simulate $\mathsf{S_2}$ signatures by relying on an EUF-CMA challenger. However, unlike condition (2d), rid^* belongs to an honest RP here, so if we make a signing query for rid^* to the EUF-CMA challenger, a signature on rid^* is not a valid forgery anymore. Thus, we do not simulate honest RP credentials with $\mathsf{S_2}$ signatures, but we simulate the NIZK proofs π's in $\mathsf{AReq_{RP}}$ oracle queries for the honest RP's without knowing/creating a valid σ_{rid}. As a result, we cannot simply rely on special soundness as in condition (2d), but we will need NIZK to be simulation extractable. Moreover, changing only RegHRP and RegCRP is not enough, but we also need to change $\mathsf{AReq_{RP}}$ so that the honest RP authentication requests can be created using the NIZK simulator. We change the oracles' behavior as follows:

$\underline{\mathsf{AInit_U}, \mathsf{ARes_{IdP}}, \mathsf{AResFin}, \mathsf{Vf_{RP}}}$**:** As they are.

$\underline{\mathsf{RegHRP}}$**:** It checks that $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$. If it holds, it updates $\mathsf{HRP} := \mathsf{HRP} \cup \{(rid, \perp)\}$ and outputs 1. If not, it outputs 0. It does not make a $\mathsf{S_2}$ signing query in any case.

RegCRP: It checks that $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$. If it does not hold, it outputs 0. Otherwise, for the registration query for rid, it queries the S_2 signing oracle, gets the signature σ_{rid}, updates $\mathsf{CRP} := \mathsf{CRP} \cup \{(rid, \sigma_{rid})\}$ on rid, and outputs σ_{rid}.

$\mathsf{AReq}_{\mathsf{RP}}$: Checks if $(rid, \cdot) \in \mathsf{HRP}$ and returns $\perp$ if not. Runs the original $O.\mathsf{AReq}_{\mathsf{RP}}$ except for computing NIZK. As we do not know a valid credential for honest rid's, the NIZK proof is simulated using the zero-knowledge simulator.

After simulating the adversary's view as above, we run the knowledge extractor for the NIZK on the proof π which corresponds to the $O.\mathsf{ARes}_{\mathsf{IdP}}$ query for $(uid, ctx^*, sid^*) \in \mathsf{Q}_\tau$ and extract a valid witness (rid, o, σ_{rid}). By the simulation extractability property of the underlying NIZK, we know that the extractor will output a valid witness with overwhelming probability, which satisfies $\mathsf{S}_2.\mathsf{Vf}(pk_2^*, rid, \sigma_{rid})$ and $\mathsf{Open}(rid, com^*, o)$.

Just as in condition (2d), if $rid \neq rid^*$, then we break the binding property of the commitment scheme using the distinct openings (rid, o) and (rid^*, o^*) to the commitment com^*. Otherwise, σ_{rid} is a valid signature on $rid^* = rid$. There is no signing oracle query to the S_2 challenger for rid^* as $rid^* \in \mathsf{HRP}$, so we output a valid forgery. We conclude that if there is a type (2e) forger adversary, the underlying NIZK is not simulation extractable, the underlying commitment scheme is not binding, or S_2 is not EUF-CMA.

□

C.4 Proof of Theorem C.1 (Req. Authentication)

THEOREM C.1 (REQUEST AUTHENTICATION). *π_{OPPID} satisfies Request Authentication if the S_2 scheme is EUF-CMA secure, and the NIZK is zero-knowledge and simulation extractable.*

Here we prove that π_{OPPID} satisfies Request Authentication (see Def. A.1) if the S_2 scheme is EUF-CMA secure, and the NIZK is zero-knowledge and simulation extractable. The proof essentially follows the RP Accountability proof from [32].

PROOF. The proof of Request Authentication follows from the zero-knowledge and simulation extractability of NIZK, and the unforgeability of S_2. The main proof strategy is similar to condition (2e) of the Session Binding proof. We aim to build a S_2 forger using an $auth$-forger, Request Authentication adversary. For that, we simulate the S_2 public key in ipk as an EUF-CMA challenge public key pk_2^*. Note that we will not actually need the full power of EUF-CMA, as we will not make any signing queries. Thus, the unforgeability of S_2 against a key-only attack, where no signing queries are allowed, would also be sufficient.

Just as in condition (2e) of the Session Binding, we do not create credentials for honest RPs, but we simulate the corresponding NIZK proofs to generate $auth$ values for these RPs. The difference from condition (2e) of the Session Binding game is that corrupted RPs are not allowed, so we do not have to simulate their credentials. Thus, we do not need to make signing queries for pk_2^* at all. In the end, we extract a valid signature on some rid value from the forged $auth$, which is a valid S_2 forgery. In more detail, we simulate the adversary's view as follows:

$\mathsf{ARes}_{\mathsf{IdP}}$: As it is.

RegHRP: It checks that $(rid, \cdot) \notin \mathsf{HRP} \cup \mathsf{CRP}$. If it holds, it updates $\mathsf{HRP} := \mathsf{HRP} \cup \{(rid, \perp)\}$ and outputs 1. If not, it outputs 0. It does not make a S_2 signing query in any case.

$\mathsf{AReq}_{\mathsf{RP}}$: Checks if $(rid, \cdot) \in \mathsf{HRP}$ and returns $\perp$ if not. Runs the original $\mathsf{AReq}_{\mathsf{RP}}$ algorithm except for computing NIZK. As we do not know a valid credential for honest rid's, the NIZK proof is simulated using the zero-knowledge simulator.

Finally, when the adversary outputs an authentication request forgery $(auth^*, crid^*, uid^*, ctx^*, sid^*)$, we run the knowledge extractor for the NIZK on the proof $auth^* := \pi^*$, and extract a valid witness (rid, o, σ_{rid}). By the winning condition, we know that there is no $O.\mathsf{AReq}_{\mathsf{RP}}$ query for $(sid^*, crid^*)$. As honest RPs bind $(sid^*, crid^*)$ to the created proofs, we know that π^* is not output by $O.\mathsf{AReq}_{\mathsf{RP}}$, so it is not a previously simulated proof by the zero-knowledge simulator. Thus, by the simulation extractability property of the underlying NIZK, (rid, o, σ_{rid}) is a valid witness with overwhelming probability, and thus $\mathsf{S}_2.\mathsf{Vf}(pk_2^*, rid, \sigma_{rid}) = 1$. As we do not make any signing queries to the unforgeability challenger, (σ_{rid}, rid) is a valid forgery against the EUF-CMA property of S_2.

□

C.5 Comparison to AIF-ZKP and OIDC

Here we provide the security comparison to OIDC and AIF-ZKP.

OIDC [43]. OIDC is the most widely deployed SSO protocol for user authentication and supports both RP authentication and RP-specific pseudonyms. According to its specification [43, §8.1], the IdP creates a pseudonym $ppid$ as $\mathsf{H}(uid||rid||k)$, where H is a cryptographic hash function and k is a secret, high-entropy random string held by the IdP. The IdP, upon receiving an authenticated request from the RP rid via the user uid, computes $ppid$ and signs both $rid, ppid$, and the session data. While this provides Request Authentication, Session Binding, and Unlinkability, OIDC clearly does not achieve Unobservability.

AIF-ZKP [32]. This work was sketched in Sec. 4.1 and provides the foundation of our protocol to achieve Session Binding, incl. strong RP authentication, and Unobservability at the same time. Our Session Binding model is stronger than in [32], and our analysis shows that the original protocol already satiesfied this stronger notion too. The original protocol reveals uid in the clear to RPs and does not support pseudonyms, i.e., it does not provide any Unlinkability.