On the Security of Rate-limited Privacy Pass

Hien Chu
Friedrich Alexander Universität Erlangen-Nürnberg
Erlangen, Germany
hien.chu@fau.de

Khue Do
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
khue.do@cispa.de

Lucjan Hanzlik
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
hanzlik@cispa.de

ABSTRACT

The privacy pass protocol allows users to redeem anonymously issued cryptographic tokens instead of solving annoying CAPTCHAs. The issuing authority verifies the credibility of the user, who can later use the pass while browsing the web using an anonymous or virtual private network. Hendrickson et al. proposed an IETF draft (privacypass-rate-limit-tokens-00) for a rate-limiting version of the privacy pass protocol, also called rate-limited Privacy Pass (RlP). Introducing a new actor called a mediator makes both versions inherently different. The mediator applies access policies to rate-limit users' access to the service while, at the same time, should be oblivious to the website/origin the user is trying to access. In this paper, we formally define the rate-limited Privacy Pass protocol and propose a game-based security model to capture the informal security notions introduced by Hendrickson et al.. We show a construction from simple building blocks that fulfills our security definitions and even allows for a post-quantum secure instantiation. Interestingly, the instantiation proposed in the IETF draft is a specific case of our construction. Thus, we can reuse the security arguments for the generic construction and show that the version used in practice is secure.

1 INTRODUCTION

Rate limiting is a widely deployed mechanism [19] that is applicable not only for paywall meters but also as protection against bots, DDoS, and credential stuffing. The typical approach is to use unique identifiers like IP addresses and geolocation data to keep track of users. Privacy-conscious users will use proxy services, VPNs, TOR networks, or other anonymization technology to protect their privacy, which makes rate limiting a challenging task.

CAPTCHAs are an alternative approach to limit bot activity, and service providers heavily rely on them in case of anonymous network users. Privacy Pass [10] was introduced as a way to decrease the burden on users in such a scenario. Users connect to an issuer to receive tokens they can later redeem instead of solving CAPTCHAs. Tokens are unlinkable to the issuing process, thus protecting users' privacy. At the same time, they assure the service provider that a valid user is connecting via the anonymous network. Unfortunately, we cannot use Privacy Pass to provide functionality comparable to rate limiting, since there exists no persistent identifier [34] of the user. Thus, it can only be seen as an excellent replacement in the CAPTCHA case. More precisely, as the issuing process is unlinkable and there are no persistent identifiers, the issuer or any other party cannot enforce rate limiting on users. Clients can also hoard tokens and use them at once. A potential solution is to keep a state for each client-origin pair to enforce limits, but keeping such a state on the issuer's side would break the client's privacy. Therefore, the idea is to introduce a new trusted party called a mediator, which will keep this state and check it against specified policies. The mediator will keep its database hidden from origins while at the same time being oblivious to the origin that the clients are trying to access.

Anonymous Submission to ACM CCS 2023, Due 19 January 2023, Copenhagen, 26-30 November 2023
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.1 Technical overview

Hendrickson et al. introduced this idea and defined a protocol called the Privacy Pass rate-limited token issuance, described in an IETF draft [20]. RlP allows service providers like Twitter and Instagram to enable metered paywall [9] and prevent abusive behavior. Besides, RlP also provides other applications such as geolocation-based policy enforcement [16, 27]. The authors propose an instantiation of the RlP protocol [20]. The protocol is claimed to be a modified Privacy Pass issuance protocol that enables tokens to be rate-limited per origin. The main difference between Privacy Pass and RlP is that in addition to the client, the issuer, and the origin, RlP introduces a new actor called a mediator. Figure 1 presents an overview of RlP with the following steps executed by each party:

The client: on receiving a challenge from the origin, creates a RlP request and later presents the token for access to the origin service.

The origin: creates a challenge for the client and later verifies the token presented by the client. It also directs the client to an appointed issuer with whom it shares its secret keys for the RlP protocol.

The mediator: verifies the authenticity of a client's request using IP address, account name, or device identifier. It later anonymizes the client's request and relays information between the client and an appointed issuer. The mediator is responsible for applying access policies defined by the origin.

The issuer: on receiving an anonymized request, issues a token on behalf of the corresponding origin. It also anonymizes the information on the origin to the mediator. Note that the issuer can be implemented as part of the origin.

A RlP protocol must ensure unforgeability, client privacy, and origin privacy. The first property states that clients can be rate-limited and cannot circumvent the mediator's policy check. Client privacy is similar to what the standard version of Privacy Pass ensures, i.e., a collaboration between the origin and issuer should be unable to link the issuance process to the client trying to gain access to the origin. On the other hand, origin privacy should hide the origin that the client is trying to access from the mediator. In their paper [20], Hendrickson et al. only draft a sketch of the abovementioned properties. Their construction is component-dependent and uses

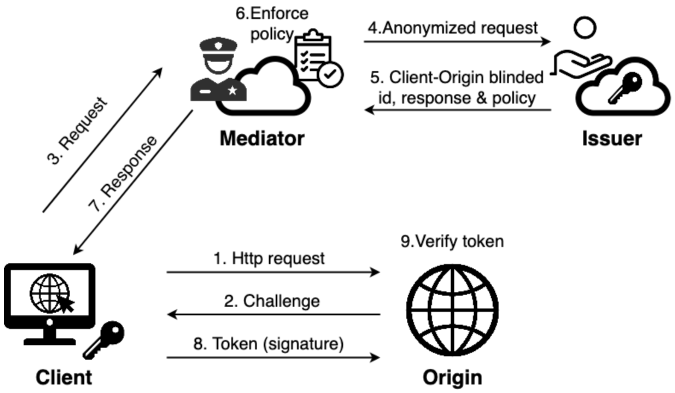

Figure 1: High-Level Protocol Overview.

specific schemes like RSA-BSSA and the key blinding property of Schnorr signatures. Unfortunately, no formal security analysis is available despite the protocol's wide adoption.

In this work, we formalize the syntax and security properties of a rate-limited Privacy Pass protocol and propose a generic construction that we show is secure in that model. The key component of our construction is a blind signature scheme playing the role of the token, while other used primitives: key encapsulation, authenticated encryption, and key blinding signatures, supporting the confidentiality of the issuing process and persistent identifier. In our scenario, blind signatures are treated as single-use and single-attribute anonymous credentials. We analyze the IETF proposed instantiation and show that it is a specific case of our generic construction, and we can easily transfer the proofs to the IETF version. As part of our analysis, we had to reformalize the key blinding signature notion to fit our needs, constituting a separate contribution that can be of independent interest. We provide a more detailed description of our contribution below.

1.2 Our contribution

Formalizing rate-limited Privacy Pass. We are the first to propose concrete syntax and security definitions that capture the requirements of RlP. We establish robust security models for privacy and unforgeability, accounting for realistic threat scenarios. We opted for simplicity while providing the adversary with as much power as possible. In particular, we allow for adaptive target selection, and instead of allowing for dynamic key corruption, we provide the adversary with all keys where it does not lead to trivial attacks. The latter allows us to define privacy without a client corruption oracle, simplifying the model.

Generic Construction. As a second contribution, we show that there exists a protocol that fulfills our security model. Our construction is modular and only depends on the security of the building blocks, which we show via standard security reduction without relying on the random oracle model or other artificial models. This approach makes a post-quantum secure instantiation possible when using post-quantum building blocks.

Reformalizing Key Blinding Signature Scheme. One building block of our generic construction is a key blinding signature scheme. This primitive was proposed in [11] and received much attention as a practical scheme [14, 33]. An essential property of this signature scheme is that a blinded public key and all signatures produced using the blinded key pair are independent of the unblinded key pair. Eaton et al. attempted to formalize the security notation of the key blinding signature scheme in [15]. Their proposed unlinkability property requires the security experiment to keep the long-term (original) public key secret. Achieving this property in real life is hard since public information tends to leak. More importantly, such a definition is incompatible with how the key blinding signature scheme is used in the RlP protocol. Motivated by this, we reformulate their model and propose a concrete security definition that can be applied to the RlP protocol. Concurrently, Eaton et al. in [13] proposed a different version for the security models of the key blinding signature scheme. At the same time, they provide a concrete security analysis for ECDSA with key blinding. In parallel, the randomizable signature scheme was proposed in [8, 17], which shares several similarities with the key-blinding signature scheme. Their security for unforgeability shares many similarities with ours. They took a different approach to the unlinkability property, in which the adversary only has access to a signing oracle but not the secret key. Interestingly, it can be shown that our indistinguishability model can capture this unlinkability notion via hybrid arguments. As part of our contribution, we show that the Schnorr signature scheme used in the IETF proposal [20] fulfills our definitions. Finally, we also develop a new proving technique that tightens the security loss for multi-agent primitives that use a key blinding signature scheme as a building block. We then use this technique to show that we can improve the tightness bounds for origin privacy under certain relaxations.

Analyzing IETF instantiation. We analyze the security of the RlP instantiation proposed in [20]. We show that this proposal is a subcase of our generic construction. Consequently, we can easily use the analysis of our construction and provide concrete bounds based on the security of well-studied component primitives. In the end, we show that two out of three security properties for this instantiation inherit a tight security reduction under strong corruption. For origin privacy, the tightness depends on a multiplicative term of the number of clients and origins in the system. We show that while relaxing our security model, we can achieve tightness linearly in the number of origins only.

2 PRELIMINARIES

Let $\lambda \in \mathbb{N}$ be the security parameter, and for integer n we define $[n] := \{1, \ldots, n\}$. We use uppercase letters $\mathcal{A}, \mathcal{B}$ to denote algorithms, and $y \leftarrow \mathcal{A}(x)$ denotes the output of $\mathcal{A}$ on input x. For a randomized algorithm $\mathcal{A}$, we use $y \xleftarrow{\$} \mathcal{A}(x)$. We write $\mathcal{A}^{\mathcal{B}}$ to denote that $\mathcal{A}$ has oracle access to $\mathcal{B}$. We say a function is negligible and denote it as $\mathsf{negl}(\lambda)$ if it vanishes faster than any polynomial.

2.1 Hybrid Public Key Encryption

We present the generic base mode of Hybrid Public Key Encryption (HPKE) in Figure 2 as in [2, 3, 25]. Below, we will define the properties of the key encapsulation mechanism, key derivation function, collision-resistant hashing, and authenticated encryption, which are the building blocks of the HPKE and are required in our analysis.

We follow the widely adopted syntax for key encapsulation mechanism found in previous works [21, 29, 31]. To define the syntax of the key derivation function and authenticated encryption, we use the notions from [25]. It is worth noting that the referenced syntax defines a key derivation function with respect to a hash function for which we give a separate definition for collision resistance. For completeness, we fully describe those building blocks in the appendix. It is worth noting that the HPKE base mode is proven to be IND-CCA-2 secure in [2, 25] while we only require IND-CCA-1 since every AEAD key will only be used one time in our constructions.

$$
\begin{array}{lcl}
\textbf{Sender} & & \textbf{Receiver} \\
 & \xleftarrow{\quad \mathsf{pk} \quad} & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.KGen}(1^\lambda) \\
(\mathsf{ct}, s) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk}) & \xrightarrow{\quad \mathsf{ct} \quad} & \\
 & & s \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{ct}) \\
 & & \mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda \\
 & & \mathsf{prk} \leftarrow \mathsf{Extract}(\mathsf{salt}, s) \\
 & & K \leftarrow \mathsf{Expand}(\mathsf{prk}, \mathsf{info}, L) \\
 & \xleftarrow{\quad c, \mathsf{salt} \quad} & c \leftarrow \mathsf{AEAD.Seal}(K, m) \\
\mathsf{prk} \leftarrow \mathsf{Extract}(\mathsf{salt}, s) & & \\
K \leftarrow \mathsf{Expand}(\mathsf{prk}, \mathsf{info}, L) & & \\
m \leftarrow \mathsf{AEAD.Open}(K, c) & &
\end{array}
$$

Figure 2: Generic HPKE base mode.

Definition 2.1 (Key Encapsulation Mechanism). A *key encapsulation mechanism* scheme $\mathsf{KEM} = (\mathsf{KEM.KGen}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$ consists of the following p.p.t. algorithm:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.KGen}(1^\lambda)$: A key generation algorithm that, on input security parameter 1^λ, outputs a pair of public key and secret key $(\mathsf{pk}, \mathsf{sk})$.
- $(\mathsf{ct}, K) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$: An encapsulation algorithm that, on input public key pk, outputs ciphertext ct and a key K.
- $\{K, \perp\} \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{ct})$: A decapsulation algorithm that, on input secret key sk and ciphertext ct, outputs key K or $\perp$ if decapsulation fails.

We say that a KEM scheme is IND-CCA secure via the advantage function $\mathbf{Adv}$ if for all λ, for any polynomial-time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KEM}}(1^\lambda) := |\Pr\left[\mathsf{IND\text{-}CCA}^{\mathcal{A}}_{\mathsf{KEM}}(1^\lambda) = 1\right] - \frac{1}{2}| \leq \mathsf{negl}(\lambda),$$

with the experiment IND-CCA defined in Figure 3.

Definition 2.2 (Key Derivation Function). A *key derivation function* $\mathsf{KDF} = (\mathsf{H}, \mathsf{Extract}, \mathsf{Expand})$ consists of the following p.p.t. algorithm:

- $c \leftarrow \mathsf{H}(\mathsf{msg})$: A hash function that, on input a message msg, outputs its N_h-byte hash value c.
- $\mathsf{prk} \leftarrow \mathsf{Extract}(\mathsf{salt}, \mathsf{km})$: An extract function that, on input random value salt and (secret) key material km, outputs a pseudorandom key prk.
- $K \leftarrow \mathsf{Expand}(\mathsf{prk}, \mathsf{info}, L)$: An Expand function that, on input pseudorandom key prk, optional context info and length L, outputs a pseudorandom key material K of length L.

Collision Resistance: Let $\mathcal{H}$ be a family of hash functions from $\{0,1\}^*$ to the finite range $\mathcal{R}$. We say that $\mathcal{H}$ is CR if for any polynomial-time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{H},\mathcal{A}}_{\mathsf{CR}} := \Pr\left[\begin{array}{c} \mathsf{H}(x_1) = \mathsf{H}(x_2) \\ \wedge x_1 \neq x_2 \end{array} \middle| \begin{array}{c} \mathsf{H} \xleftarrow{\$} \mathcal{H} \\ (x_1, x_2) \leftarrow \mathcal{A}^{\mathsf{H}} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Multi-key Pseudorandom Function: Let $F : \mathcal{K} \times \{0,1\}^* \to \mathcal{R}$ be a keyed function with a finite key space $\mathcal{K}$ and finite output range $\mathcal{R}$. We say that F is a (n, q)-PRF if for any polynomial-time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{F,\mathcal{A}}_{(n,q)\text{-}\mathsf{PRF}} := \left|\Pr[\mathcal{A}^{f_1,\ldots,f_n}] - \Pr_{K_1,\ldots K_n \xleftarrow{\$} \mathcal{K}}[\mathcal{A}^{F(K_1,\cdot),\ldots,F(K_n,\cdot)}]\right| \leq \mathsf{negl}(\lambda),$$

where $f_i : \{0,1\}^* \to \mathcal{R}$ for $i \in [n]$ are perfect random functions and $\mathcal{A}$ makes at most q queries in total to the oracle f_i.

For a key derivation function KDF defined as above, we bound the advantage of KDF by:

$$\mathbf{Adv}^{\mathsf{KDF},\mathcal{A}}_{(q,q)\text{-}\mathsf{PRF}} \leq \mathbf{Adv}^{\mathsf{Extract},\mathcal{A}_1}_{\mathsf{CR}} + \mathbf{Adv}^{\mathsf{Extract},\mathcal{A}_2}_{(q,q)\text{-}\mathsf{PRF}} + \mathbf{Adv}^{\mathsf{Expand},\mathcal{A}_3}_{(q,q)\text{-}\mathsf{PRF}}.$$

Definition 2.3 (Authenticated Encryption with Associated Data). An *authenticated encryption with associated data scheme* $\mathsf{AEAD} = (\mathsf{AEAD.Seal}, \mathsf{AEAD.Open})$ consists of the following p.p.t. algorithm:

- $\mathsf{ct} \leftarrow \mathsf{AEAD.Seal}(\mathsf{sk}, n, \mathsf{aad}, \mathsf{msg})$ An encryption algorithm that, on input secret key sk, nonce n, optional addition data aad and message msg, outputs a corresponding ciphertext ct.
- $\mathsf{msg} \leftarrow \mathsf{AEAD.Open}(\mathsf{sk}, n, \mathsf{aad}, \mathsf{ct})$ A decryption algorithm that, on input secret key sk, nonce n, optional addition data aad and ciphertext ct under the key sk, outputs the corresponding plaintext msg or $\perp$ if decryption fails.

We say that an AEAD scheme is IND-CCA secure if for all λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{AEAD}}(1^\lambda) := \left|\Pr\left[\mathsf{IND\text{-}CCA}^{\mathcal{A}}_{\mathsf{AEAD}}(1^\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

with the experiment IND-CCA defined in Figure 3.

2.2 Blind Signature Scheme

Definition 2.4 (2-move Blind Signature Scheme). A *2-move blind signature scheme* $\mathsf{BS} = (\mathsf{BS.KGen}, \mathsf{BS.Blind}, \mathsf{BS.PreSig}, \mathsf{BS.Ext}, \mathsf{BS.Vf})$ consists of the following p.p.t. algorithms:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KGen}(1^\lambda)$: The *key generation protocol* KGen outputs public key pk and secret key sk.
- $(\mathsf{msg}', \mathsf{aux}) \leftarrow \mathsf{BS.Blind}(\mathsf{pk}, \mathsf{msg})$: The *blinding* is a probabilistic algorithm that, on input the public key pk and a string msg, outputs a blinded message msg′ and an auxiliary string aux.

$\mathsf{IND\text{-}CCA}^{\mathcal{A}}_{\mathsf{KEM}}(1^\lambda)$	$\mathsf{IND\text{-}CCA}^{\mathcal{A}}_{\mathsf{AEAD}}(1^\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KEM.KGen}(1^\lambda)$	$\mathsf{sk} \xleftarrow{\$} \{0,1\}^\lambda$
$b \xleftarrow{\$} \{0,1\}$	$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{ct}, K_0) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$	$(\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}_1^{\mathsf{AEAD.Open}_{\mathsf{ct}}}$
$K_1 \xleftarrow{\$} \mathcal{K}$	$\mathsf{ct} \leftarrow \mathsf{AEAD.Seal}(\mathsf{sk}, n, \mathsf{aad}, \mathsf{msg}_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{KEM.Dec}_{\mathsf{ct}}}(\mathsf{pk}, \mathsf{ct}, K_b)$	$b' \leftarrow \mathcal{A}_2^{\mathsf{AEAD.Open}_{\mathsf{ct}}}(\mathsf{ct})$
return $b' = b$	**return** $b' = b$
$\mathsf{KEM.Dec}_{\mathsf{ct}}(\mathsf{ct}')$	$\mathsf{AEAD.Open}_{\mathsf{ct}}(\mathsf{ct}')$
if $\mathsf{ct}' = \mathsf{ct}$ **then** : **return** $\perp$	**if** $\mathsf{ct}' = \mathsf{ct}$ **then** : **return** $\perp$
$K \leftarrow \mathsf{KEM.Dec}(\mathsf{sk}, \mathsf{ct}')$	$K \leftarrow \mathsf{AEAD.Open}(\mathsf{sk}, n, \mathsf{aad}, \mathsf{msg}')$
return K	**return** msg

Figure 3: Experiments for HPKE.

$\sigma' \leftarrow \mathsf{BS.PreSig}(\mathsf{sk}, \mathsf{msg}')$: The *pre-signing* algorithm BS.PreSig is run by the signer using secret key sk and blinded message msg' in order to produce a pre-signature σ'.

$\sigma \leftarrow \mathsf{BS.Ext}(\sigma', \mathsf{aux})$: The *extracting* algorithm is run by the user to convert a pre-signature σ' to a final signature σ.

$\{0,1\} \leftarrow \mathsf{BS.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$: The *verification* algorithm takes as input a public key pk, a message m, and an extracted signature σ. It outputs a bit b, indicating the validity of the signature.

We say that a BS scheme is correct if for every λ, for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KB.KGen}(1^\lambda)$, and for every message msg, the following holds:

$$\Pr\left[\ \mathsf{BS.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma) = 1 \ \middle|\ \begin{array}{c} (\mathsf{msg}', \mathsf{aux}) \leftarrow \mathsf{BS.Blind}(\mathsf{pk}, \mathsf{msg}) \\ \sigma' \leftarrow \mathsf{BS.PreSig}(\mathsf{sk}, \mathsf{msg}') \\ \sigma \leftarrow \mathsf{BS.Ext}(\sigma', \mathsf{aux}) \end{array} \right] = 1.$$

Definition 2.5 (Blindness). A blind signature scheme BS is blind if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all 1^λ and all PPT adversaries $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Blind}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Blind}}(1^\lambda) = |\Pr\left[\mathsf{ExpBlind}^{\mathcal{A}}_{\mathsf{BS}}(1^\lambda) = 1\right] - \frac{1}{2}|$ and the experiment $\mathsf{ExpBlind}^{\mathcal{A}}_{\mathsf{BS}}$ is defined in Figure 4.

Definition 2.6 (One-more unforgeability). A blind signature BS is one-more unforgeable if there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all 1^λ and all PPT adversaries $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{BS.OMUF}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{BS.OMUF}}(1^\lambda) = \Pr\left[\mathsf{ExpOUnf}^{\mathcal{A}}_{\mathsf{BS}}(1^\lambda) = 1\right]$ and the experiment $\mathsf{ExpOUnf}^{\mathcal{A}}_{\mathsf{BS}}$ is defined in Figure 5. Note that ℓ is the number of queries to the BS.PreSig oracle.

$\mathsf{ExpBlind}^{\mathcal{A}}_{\mathsf{BS}}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{msg}_0, \mathsf{msg}_1, \mathsf{Db}) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{msg}'_i, \mathsf{aux}_i) \leftarrow \mathsf{BS.Blind}(\mathsf{pk}, \mathsf{msg}_i)$ **for** $i = b, 1-b$
$(\sigma'_b, \sigma'_{1-b}) \leftarrow \mathcal{A}(\mathsf{msg}'_b, \mathsf{msg}'_{1-b})$
$\sigma_i \leftarrow \mathsf{BS.Ext}(\sigma'_i, \mathsf{aux}_i)$ **for** $i = b, 1-b$
If $\mathsf{BS.Vf}(\mathsf{pk}, \mathsf{msg}_0, \sigma_0) = 0$ **or** $\mathsf{BS.Vf}(\mathsf{pk}, \mathsf{msg}_1, \sigma_1) = 0$ **then** :
$\quad (\sigma_0, \sigma_1) = (\perp, \perp)$
$b' \leftarrow \mathcal{A}(\mathsf{Db}, \sigma_0, \sigma_1)$
return $b = b'$

Figure 4: Blindness experiment $\mathsf{ExpBlind}^{\mathcal{A}}_{\mathsf{BS}}$**.**

$\mathsf{ExpOUnf}^{\mathcal{A}}_{\mathsf{BS}}(1^\lambda)$	$\mathsf{PreSig}(\mathsf{msg})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{BS.KGen}(1^\lambda)$	$\ell \leftarrow \ell + 1$
$\ell \leftarrow 0$	**return** $\mathsf{BS.PreSig}(\mathsf{sk}, \mathsf{msg})$
$\{(\mathsf{msg}_i, \sigma_i)\}_{i=1}^{\ell+1} \leftarrow \mathcal{A}^{\mathsf{PreSig}}(\mathsf{pk})$	
$b_1 \leftarrow \wedge_{i=1}^{\ell+1} \mathsf{BS.Vf}(\mathsf{pk}, \mathsf{msg}_i, \sigma_i)$	
$b_2 \leftarrow \mathsf{msg}_i \neq \mathsf{msg}_j \forall 1 \leq i < j \leq \ell + 1$	
return $b = b_1 \wedge b_2$	

Figure 5: Unforgeability experiment $\mathsf{ExpOUnf}^{\mathcal{A}}_{\mathsf{BS}}$**.**

3 KEY-BLINDING SIGNATURE SCHEMES

We formalize the notion of key-blinding signature schemes and the corresponding security notions. Intuitively, this scheme enables signers to blind their private signing key, generating each signature using a signing key and blinding key independent of the signing key. The scheme is required to be unforgeable and indistinguishable.

Definition 3.1 (Key Blinding Signature Schemes). A *key blinding signature scheme* KB = (KB.KGen, KB.Sign, KB.Vf, KB.PKBlind, KB.SKBlind, KB.BlSign, KB.Unblind) consists of the following p.p.t. algorithm:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KB.KGen}(1^\lambda)$: A key generation algorithm that, on input security parameter 1^λ, defines a key space κ and outputs a pair of public and secret key (pk, sk).

$\sigma \leftarrow \mathsf{KB.Sign}(\mathsf{sk}, \mathsf{msg})$: A signing algorithm that on input secret key sk and a message msg, outputs the signature σ.

$\{0,1\} \leftarrow \mathsf{KB.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$: A verification algorithm that, on input the public key pk, a message msg and a signature σ, outputs 1 if σ is a valid signature for msg and 0 otherwise.

$\{\mathsf{pk}', \perp\} \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk})$: A public key blinding algorithm that, on input long-term public key pk and a blind key bk, outputs a blinded public key pk' if $\mathsf{pk}' \in \kappa$ and aborts otherwise.

$\{\mathsf{pk}', \perp\} \leftarrow \mathsf{KB.SKBlind}(\mathsf{sk}, \mathsf{bk})$: A private key blinding algorithm that, on input secret key pk and a blind key bk, outputs a blinded public key pk' if $\mathsf{pk}' \in \kappa$ and aborts otherwise.

$\sigma \leftarrow$ KB.BlSign(sk, bk, msg): An adaptive signing algorithm that, on input secret key sk, a blind key bk and a message msg, outputs the corresponding signature σ.

$\{\mathsf{pk}, \perp\} \leftarrow$ KB.Unblind(pk′, bk): A key unblind algorithm that, on input blinded public key pk′ and its corresponding blind key bk, outputs the long term public key pk if such public key exists and aborts otherwise.

We say that a KB scheme is correct if for every λ, for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KB.KGen}(1^\lambda)$, and for every message msg, the following holds:

$$\Pr\left[\mathsf{KB.Vf}(\mathsf{pk}', \mathsf{msg}, \sigma) = 1 \;\middle|\; \begin{array}{c} \mathsf{bk} \xleftarrow{\$} \{0,1\}^\lambda \\ \mathsf{pk}' \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk}) \\ \sigma \leftarrow \mathsf{KB.BlSign}(\mathsf{sk}, \mathsf{bk}, \mathsf{msg}) \end{array}\right] = 1.$$

We say that a KB scheme has double key blinding if for every λ, and for every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KB.KGen}(1^\lambda)$, the following holds:

$$\Pr\left[\begin{array}{c} \mathsf{KB.Unblind}(\mathsf{pk}_2, \mathsf{bk}_1) \\ = \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk}_2) \end{array} \;\middle|\; \begin{array}{c} \mathsf{bk}_1, \mathsf{bk}_2 \xleftarrow{\$} \{0,1\}^\lambda \\ \mathsf{pk}_1 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk}_1) \\ \mathsf{pk}_2 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}_1, \mathsf{bk}_2) \end{array}\right] = 1.$$

By inductivity, it is clear that any double-key blinding scheme would achieve multiple times key blinding. The two properties above hold for both public and private key blinding.

Definition 3.2 (Indistinguishability). We say that a KB scheme is indistinguishable if for all λ, for any polynomial-time $\mathcal{A}$, given the key pair (sk, pk), the following holds: $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KBlind}}(1^\lambda) \leq \mathsf{negl}(\lambda)$, where

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KBlind}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpKBlind}^{\mathcal{A}}_{\mathsf{KB}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$$

with the experiment ExpKBlind, where we require the adversary to distinguish between a freshly generated key and a blinded key from the long-term key, defined in Figure 6. When $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KBlind}}(1^\lambda) = 0$, we say that the key-blinding signature scheme KB has perfect indistinguishability. We also consider a different version of the ExpKBlind experiment, denoted as ExpBrKBlind, where not only do we not give the secret key sk* to the adversary, but the requirement for the experiment is also different in that we ask the adversary to distinguish if the same blind key bk* is re-used. In this experiment, given only the public pk*, the adversary, however, has access to the oracle BlSign and KBlind. The reason is that exposing the secret key sk* to the adversary would make this game trivial since the adversary could query the KBlind oracle with this key. The advantage $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{BrKBlind}}(1^\lambda)$ is defined similarly.

Definition 3.3 (Unforgeability). We say that a KB scheme has unforgeability if for all λ, for any polynomial time algorithm $\mathcal{A}$, given only the public key pk and access to BlSign oracle, the following holds: $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KB.UF}}(1^\lambda) \leq \mathsf{negl}(\lambda)$, where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{KB.UF}}(1^\lambda) = \Pr\left[\mathsf{ExpUF}^{\mathcal{A}}_{\mathsf{KB}}(1^\lambda) = 1\right]$ with ExpUF defined in Figure 7.

Next, we present a concrete instantiation for the key blinding signature scheme. We follow the constructions proposed in [11, 26] for the Schnorr signature scheme with key blinding functionality. It can be shown that this scheme achieves all the desired properties of a key blinding signature scheme. Due to space reasons, we postpone a formal proof to the full version of the paper.

$\underline{\mathsf{ExpKBlind}^{\mathcal{A}}_{\mathsf{KB}}(1^\lambda)}$	$\underline{\mathsf{ExpBrKBlind}^{\mathcal{A}}_{\mathsf{KB}}(1^\lambda)}$
$(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KB.KGen}(1^\lambda)$	$(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KB.KGen}(1^\lambda)$
$\mathsf{msg}^* \leftarrow \mathcal{A}(\mathsf{pk}^*, \mathsf{sk}^*)$	$\mathsf{bk}^* \xleftarrow{\$} \{0,1\}^\lambda$
$\mathsf{bk}^* \xleftarrow{\$} \{0,1\}^\lambda$	$\mathsf{pk}_0 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}^*, \mathsf{bk}^*)$
$\mathsf{pk}_0 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}^*, \mathsf{bk}^*)$	$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{KB.KGen}(1^\lambda)$
$(\mathsf{sk}_1, \mathsf{pk}_1) \xleftarrow{\$} \mathsf{KB.KGen}(1^\lambda, \kappa)$	$b \xleftarrow{\$} \{0,1\}$
$b \xleftarrow{\$} \{0,1\}$	$b' \leftarrow \mathcal{A}^{\mathsf{BlSign},\mathsf{KBlind}}(\mathsf{pk}_b, \mathsf{pk}^*)$
$\sigma_0 \leftarrow \mathsf{KB.BlSign}(\mathsf{sk}^*, \mathsf{bk}^*, \mathsf{msg}^*)$	**return** $b = b'$
$\sigma_1 \leftarrow \mathsf{KB.Sign}(\mathsf{sk}_1, \mathsf{msg}^*)$	
$b' \leftarrow \mathcal{A}(\mathsf{pk}_b, \sigma_b)$	$\underline{\mathsf{BlSign}(\mathsf{bk}, \mathsf{msg}')}$
return $b = b'$	$\sigma' \leftarrow \mathsf{KB.BlSign}(\mathsf{sk}^*, \mathsf{bk}, \mathsf{msg}')$
	$\underline{\mathsf{KBlind}(\mathsf{sk})}$
	$\mathsf{pk}' \leftarrow \mathsf{KB.SKBlind}(\mathsf{sk}, \mathsf{bk}^*)$
	return pk'

Figure 6: Experiments for Indistinguishability of a KB Scheme.

$\underline{\mathsf{ExpUF}^{\mathcal{A}}_{\mathsf{KB}}(1^\lambda)}$	$\underline{\mathsf{BlSign}(\mathsf{bk}, \mathsf{msg}')}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KB.KGen}(1^\lambda)$	**if** $\mathsf{msg}' \in \mathcal{M}$ **then** : **Abort**
$(\mathsf{pk}', \mathsf{bk}, \mathsf{msg}, \sigma) \leftarrow \mathcal{A}^{\mathsf{BlSign}}(\mathsf{pk})$	$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathsf{msg}'\}$
$b_1 \leftarrow \mathsf{KB.Vf}(\mathsf{pk}', \mathsf{msg}, \sigma)$	$\sigma' \leftarrow \mathsf{KB.BlSign}(\mathsf{sk}, \mathsf{bk}, \mathsf{msg}')$
$b_2 \leftarrow \mathsf{pk} = \mathsf{KB.Unblind}(\mathsf{pk}', \mathsf{bk})$	**return** σ'
$b_3 \leftarrow \mathsf{msg} \notin \mathcal{M}$	
return $b = b_1 \wedge b_2 \wedge b_3$	

Figure 7: Experiments for Unforgeability of a KB Scheme.

Definition 3.4 (Key Blinding Schnorr Signature Scheme). Consider a group $\mathbb{G}$ of prime order q, generated by g, and a hash function H. A *key blinding Schnorr signature scheme* KB = (KB.KGen, KB.Sign , KB.Vf, KB.PKBlind, KB.SKBlind, KB.BlSign, KB.Unblind) consists of the following p.p.t. algorithm:

$(h, s) \leftarrow \mathsf{KB.KGen}(1^\lambda)$: The key generation algorithm first defines the key space κ to be the group $\mathbb{G}$ excluding the identity element. It then randomly generates the secret key $s \xleftarrow{\$} \mathbb{Z}_q$ and computes $h \leftarrow g^s$. The public key pk is the pair (g, h).

$(c, z) \leftarrow \mathsf{KB.Sign}(s, \mathsf{msg})$: The signing algorithm first generates a random nonce $r \xleftarrow{\$} \mathbb{Z}_q$ and computes g^r. It then computes $c \leftarrow \mathsf{H}(\mathsf{pk}, g^r, \mathsf{msg})$ where H is a hash function and $z \leftarrow r - c \cdot s \mod q$. The algorithm outputs the pair (c, z) as the signature σ.

$\{0,1\} \leftarrow \mathsf{KB.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$: The verification algorithm extracts the information from the signature $(c, z) := \sigma$ and the public key $(g, h) := \mathsf{pk}$. It then computes $R_v \leftarrow g^z \cdot h^c$ and $c_v \leftarrow \mathsf{H}(\mathsf{pk}, R_v, \mathsf{msg})$. The algorithm outputs 1 if $c = c_v$ and 0 otherwise.

$\{(g, h'), \perp\} \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk})$: The public key blinding algorithm extracts the information from the public key $(g, h) := \mathsf{pk}$. It then computes $h' \leftarrow h^{\mathsf{bk}}$ and outputs the pair (g, h') as the blinded public key pk' if $\mathsf{pk}' \in \kappa$ and aborts otherwise.

$\{(g, h'), \perp\} \leftarrow \mathsf{KB.SKBlind}(\mathsf{sk}, \mathsf{bk})$: The private key blinding algorithm computes $h' \leftarrow g^{\mathsf{sk} \cdot \mathsf{bk}}$ and outputs the pair (g, h') as the blinded public key pk' if $\mathsf{pk}' \in \kappa$ and aborts otherwise.

$(c, z) \leftarrow \mathsf{KB.BlSign}(\mathsf{sk}, \mathsf{bk}, \mathsf{msg})$: The adaptive signing algorithm executes the algorithm $(c, z) \leftarrow \mathsf{KB.Sign}(\mathsf{sk} \cdot \mathsf{bk}, \mathsf{msg})$ and outputs the signature $\sigma = (c, z)$.

$\mathsf{pk} \leftarrow \mathsf{KB.Unblind}(\mathsf{pk}', \mathsf{bk})$: The key unblinding algorithm extracts the information from the public key $(g, h') := \mathsf{pk}'$. It then derives the unblinded public key by computing $h \leftarrow h'^{\mathsf{bk}^{-1}}$. The algorithm outputs the pair (g, h) as the unblinded public key pk if $\mathsf{pk} \in \kappa$ and aborts otherwise.

Theorem 3.5. *The key blinding Schnorr signature scheme defined in 3.4 is correct, double key blinding, achieving unforgeability and indistinguishability.*

4 RATE-LIMITED PRIVACY PASS

This section will define a rate-limited Privacy Pass (RlP) protocol and its security properties. The RlP protocol begins with the client sending an HTTP request to the origin as a request to gain access to the origin-provided content. In response, the origin replies with a challenge ch. In the next step, the client executes the $\mathsf{RlP.Rq}(C_{\mathsf{sk}}, O_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})$ algorithm to create a request rq and some auxiliary opening information aux that it stores for later use. The keys $C_{\mathsf{sk}}, O_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}$ are, respectively, the client's secret key, the origin's public key, and the issuer's public key. The keys are generated using respective key generation algorithms RlP.CSetup, RlP.OSetup, and RlP.ISetup. We use a static set of origins generated at once by RlP.OSetup. We opted for this approach since it simplifies modeling the key generation of the issuer, i.e., RlP.ISetup takes as input the secret of all origins at once. In a real-world implementation, the number of origins will be dynamic, and so will the the secret key of the issuer. It is also worth noting that a trusted party generates the secret keys of clients, and we model this by using a single key generation RlP.CSetup. The main reason behind this is that the secret keys are used to generate user-origin-specific identifiers that the mediator uses for rate limiting. User-generated keys would allow the clients to circumvent the mediator. For a practical example, one can think of the client's secret keys to be generated in a secure enclave of the user's device. In the next step, the client forwards the request rq and its public key C_{pk} to the mediator, who uses the $\mathsf{RlP.FwdRq}(C_{\mathsf{pk}}, \mathsf{rq})$ algorithm to create an anonymized request rq′ sent to the issuer. For each request, the issuer executes the $\mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}^{n_O}_{j=1}, \mathsf{rq}')$ algorithm using its secret key $\mathcal{I}_{\mathsf{sk}}$ and all the secret keys and secret identities of origins. Note that this is required since the issuer cannot distinguish the origin to which the request corresponds. Otherwise, the mediator could also link the client's request to an origin. The algorithm outputs an anonymous session index id, the response re, and policy predicate function f, which the issuer forwards to the mediator. The mediator now checks whether the client fulfills the policy f. To do so, it executes the $\mathsf{RlP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$ algorithm, which outputs the response or $\perp$. The mediator keeps an internal database state Db, which contains client/origin-specific entries. Clients' privacy is preserved using anonymous identifiers (i.e., id). Before forwarding re to the client, the mediator updates its database using $\mathsf{RlP.Update}(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$. Finally, the client redeems the response using the $\mathsf{RlP.Rdm}(O_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux})$ algorithm and forwards the output token σ and the challenge ch to the origin. The origin grants access to the content for the client if $\mathsf{RlP.Vf}(O_{\mathsf{pk}}, \mathsf{ch}, \sigma)$ outputs 1.

A detailed representation of the protocol can be found in Figure 8. We give a more formal description of the syntax below.

4.1 Protocol Syntax

Definition 4.1 (Rate-limited Privacy Pass (RlP)). A *rate-limited Privacy Pass* RlP consists of the following p.p.t. algorithms:

$(\{C^i_{\mathsf{sk}}\ C^i_{\mathsf{pk}}\}^{n_C}_{i=1}) \leftarrow \mathsf{RlP.CSetup}(1^\lambda, n_C)$: A client setup algorithm that, on input security parameter 1^λ and the number of clients n_C, outputs public and secret key-pairs for each client C^i.

$(\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}^{n_O}_{j=1}) \leftarrow \mathsf{RlP.OSetup}(1^\lambda, n_O)$: An origin setup algorithm that, on input security parameter 1^λ and the number of origins n_O, outputs public and secret key-pairs for each origin O^j.

$(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{id}}\}^{n_O}_{j=1}) \leftarrow \mathsf{RlP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}^{n_O}_{j=1})$: A setup algorithm that, on input security parameter 1^λ and public key O^j_{pk} of each origin, outputs public and secret key-pair of the Issuer $\mathcal{I}$ and period secret ids O^j_{id} of origins.

$(\mathsf{rq}, \mathsf{aux}) \leftarrow \mathsf{RlP.Rq}(C^i_{\mathsf{sk}}, O^j_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})$: A request algorithm that, on input client secret key C^i_{sk}, origin public key O^j_{pk} issuer public key$\mathcal{I}_{\mathsf{pk}}$, and the challenge ch from origin O^j, outputs a request rq.

$\mathsf{rq}' \leftarrow \mathsf{RlP.FwdRq}(C^i_{\mathsf{pk}}, \mathsf{rq})$: A proxy request algorithm that, on input request rq, outputs an anonymized request rq′ when rq is a valid request and $\perp$ otherwise.

$\{(\mathsf{id}, \mathsf{re}, f), \perp\} \leftarrow \mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}},\}^{n_O}_{j=1}, \mathsf{rq}')$: An issue algorithm that, on input issuer secret key $\mathcal{I}_{\mathsf{sk}}$, all origin secret key O^j_{sk} and origin secret id O^j_{id} for $j \in \{1, \ldots, n_O\}$, and an anonymized request rq′, outputs an anonymous index id, response re and policy predicate function f and $\perp$ otherwise.

$\{\mathsf{re}, \perp\} \leftarrow \mathsf{RlP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$: A policy enforcement algorithm that, on input anonymous index id, response re, the mediator's database Db and the policy predicate function f, outputs response re if $f(\mathsf{id}, \mathsf{rq}, \mathsf{Db}) = 1$ and $\perp$ otherwise.

$\{\sigma, \perp\} \leftarrow \mathsf{RlP.Rdm}(O^j_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux})$: A token redeem algorithm that, on input public key O^j_{pk}, the response re and the challenge ch, outputs the token σ and $\perp$ otherwise.

$\{0, 1\} \leftarrow \mathsf{RlP.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}, \sigma)$: A verification algorithm that, on input origin public key O^j_{pk}, the challenge ch and token σ, outputs 1 if σ is the valid token and 0 otherwise.

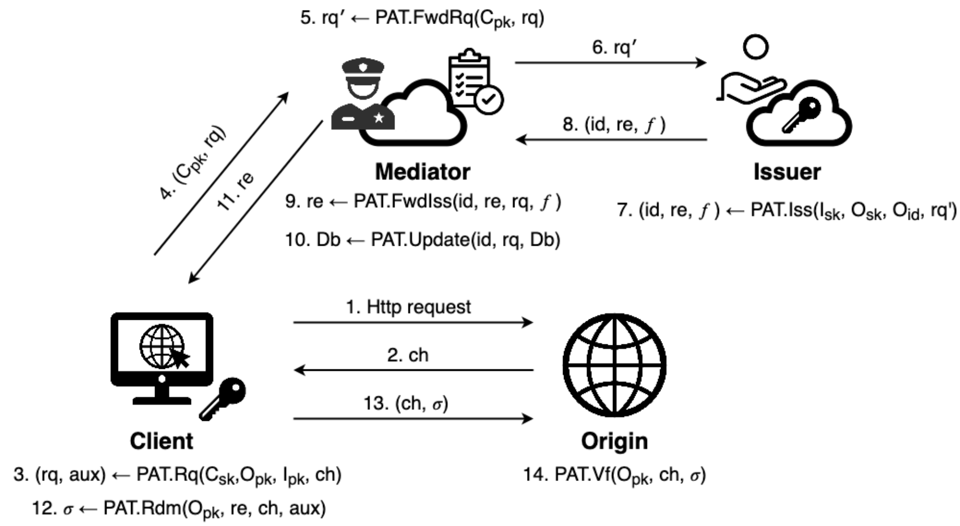

Figure 8: Visual representation of our PAT syntax, where we clarify the algorithms executed by each party.

Db ← RlP.Update(id, rq, Db): An update algorithm that, on input a session index id, the corresponding request rq and the database Db, outputs an updated database Db.

We say that a RlP scheme is correct if for all λ, for every $(\{C^i_{\mathsf{sk}}\, C^i_{\mathsf{pk}}\} \leftarrow \mathsf{RlP.CSetup}(1^\lambda, n_C)$, for every $(\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}^{n_O}_{j=1}) \leftarrow \mathsf{RlP.OSetup}(1^\lambda, n_O)$, for every $(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{id}}\}^{n_O}_{j=1}) \leftarrow \mathsf{RlP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}^{n_O}_{i=1})$, every $\mathsf{ch} \xleftarrow{\$} \{0,1\}^\lambda$ and $i \in [nc], k \in [n_O]$, the following is 1:

$$\Pr\left[\begin{array}{c|c} & (\mathsf{rq}, \mathsf{aux}) \leftarrow \mathsf{RlP.Rq}(C^i_{\mathsf{sk}}, O^k_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch}) \\ \mathsf{RlP.Vf}(O^k_{\mathsf{pk}}, \mathsf{ch}, \sigma) & (\mathsf{id}, \mathsf{re}, f) \leftarrow \mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}} O^j_{\mathsf{id}}\}^{n_O}_{j=1}, \mathsf{rq}') \\ = 1 & \sigma \leftarrow \mathsf{RlP.Rdm}(O^k_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux}) \end{array}\right]$$

4.2 Security and Privacy Models

To simplify the experiment description, our security model assumes a multi-stage adversary $\mathcal{A}$ with an implicit state. In other words, we use the same notation $\mathcal{A}$ to indicate the adversary in different stages and do not make the private state of $\mathcal{A}$ explicit. We also model a static case where all keys are computed in a single setup algorithm. More precisely, all the client's keys are pre-computed and then distributed to the clients. This prevents the clients from generating keys ad-hoc without a trusted setup, which could later break the rate-limiting protection.

Definition 4.2 (Oracle). Let $\mathcal{A}$ be a stateful polynomial time adversary and RlP be a rate-limited Privacy Pass. In our security experiments we give the adversary $\mathcal{A}$ access to the following oracles Rq, Iss, Rdm, FwdRq, FwdIss, and Update defined as follows:

- $\mathsf{Rq}(i, k, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})$: This oracle executes a RlP.Rq algorithm for client C^i, origin O^k and issuer $\mathcal{I}$ on the challenge ch. In some experiments, we will limit the access only to clients C^i and origins O^j that are not the targets of the adversary $\mathcal{A}$.
- Iss(rq): This oracle executes an issue algorithm RlP.Iss on a given request rq.
- $\mathsf{Rdm}(k, \mathsf{re}', \mathsf{ch}, \mathsf{aux})$: This oracle executes a token redeem algorithm RlP.Rdm for a given response re′, challenge ch and auxiliary information aux.
- FwdRq(rq): This oracle executes a forward request algorithm RlP.FwdRq for request rq.
- $\mathsf{FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$: This oracle executes a forward issue algorithm RlP.FwdIss for session index id, response re, request rq, database Db and policy f.
- Update(id, rq, Db): This oracle can be used to either arbitrarily modify the database Db or execute update algorithm RlP.Update for session index id, request rq and database Db.

4.2.1 *Unforgeability.*

Unforgeability ensures that no client can obtain a valid token from anywhere except for the successful RlP execution. Informally, we model the case where the client is the adversary who wants to bypass the policy, while the issuer and the mediator are trusted. We define unforgeability as a one-more experiment between an adversary $\mathcal{A}$ acting as a client and a challenger simulating the honest mediator and issuer. The description follows.

Definition 4.3 (Unforgeability). For a RlP protocol and a stateful, polynomial time adversary $\mathcal{A}$, we define the one-more unforgeability experiment as follows:

- **Setup phase.** The adversary receives an issuer and an origin public key from the challenger. In addition, it receives a list of n_C public keys $\{C^i_{\mathsf{pk}}\}^{n_C}_{i=1}$ of all authenticated clients. The adversary then outputs a target public key, which the mediator then authenticates and adds to the list as $C^{i^*}_{\mathsf{pk}}$. Conventionally, $i^* = n_C + 1$ and $C^{i^*}_{\mathsf{pk}}$ must be distinct from previous client public keys.
- **Query phase.** The adversary is given access to two oracles. The first oracle is the token challenge query to the origin O. The second oracle provides the adversarial client with a way to execute the RlP protocol with the mediator and the issuer. Concretely, at each query, the adversary prepares a request rq, upon which the

oracle answers either re or $\perp$ based on the following computation.

$$\mathsf{rq}' \leftarrow \mathsf{RIP.FwdRq}(C^i_{\mathsf{pk}}, \mathsf{rq})$$
$$\{(\mathsf{id}, \mathsf{re}, f), \perp\} \leftarrow \mathsf{RIP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}^{n_O}_{j=1}, \mathsf{rq}')$$
$$(\{\mathsf{re}, \perp\}) \leftarrow \mathsf{RIP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$$

Let ℓ be the number of successful queries to the second oracle, meaning that re $\neq \perp$. At the end of this phase, let O^j be the target origin, and the challenger updates the database Db so that C^{i^*} is at the policy limit while freeing the others.

- **Output phase.** The adversary $\mathcal{A}$ outputs $\ell + 1$ final tokens $\{\mathsf{ch}_k, \sigma_k\}^{\ell+1}_{k=1}$.

The adversary wins if all $\ell + 1$ final tokens are valid.

We say that a RIP scheme is unforgeable if for all λ and any public key O_{pk}, for any polynomial time algorithm $\mathcal{A}$, given only O_{pk}, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OMUF}} \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OMUF}} = \Pr\left[\prod^{l+1}_{k=1} \mathsf{RIP.Vf}(O_{\mathsf{pk}}, \sigma_k, \mathsf{ch}_k) = 1\right]$.

4.2.2 Client Privacy. Informally, we model the case where the mediator is the only trusted party, while the issuer and the origin are considered to be malicious. We also let all the client's keys be exposed to the adversary. We want to ensure that a collaboration between a malicious issuer and origin cannot link any client with a corresponding RIP issuance execution. Any metadata exchanged outside the protocol is considered out of scope.

Definition 4.4 (Client Privacy). For RIP and a stateful, polynomial adversary $\mathcal{A}$, we define the experiment $\mathsf{ExpCP}^{\mathcal{A}}_{\mathsf{RIP}}$ as follows:

- **Setup phase.** The adversary begins by outputting one issuer public key and n_O origin public keys. The challenger creates honest key pairs for a set of n_C clients but allows the adversary to access the entire set, even client secret keys.
- **Query phase.** The adversary is free to query the oracles FwdRq, FwdIss before outputting two token challenges $\mathsf{ch}_0, \mathsf{ch}_1$, a target origin public key $O^{j^*}_{\mathsf{pk}}$ and two target client public keys $C^{i_0}_{\mathsf{pk}}, C^{i_1}_{\mathsf{pk}}$.
- **Challenge phase.** The challenger executes two honest clients and one mediator, and the adversary $\mathcal{A}$ engages in two executions with them. The challenger computes $\tilde{\mathsf{rq}}_k \leftarrow \mathsf{RIP.FwdRq}(\mathsf{rq}_k)$ where $\mathsf{rq}_k \leftarrow \mathsf{RIP.Rq}(C^{i_k}_{\mathsf{sk}}, O^{j^*}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch}_k)$ for $k = 0, 1$. The challenger then sends the anonymized requests $\tilde{\mathsf{rq}}_b, \tilde{\mathsf{rq}}_{1-b}$ to the adversary $\mathcal{A}$ for some bit $b \xleftarrow{\$} \{0, 1\}$. Next, the adversary $\mathcal{A}$ sends back $\mathsf{re}_b, \mathsf{re}_{1-b}$. The challenger then computes $(\sigma_k) \leftarrow \mathsf{RIP.Rdm}(O^{j^*}_{\mathsf{pk}}, \tilde{\mathsf{re}}_k, \mathsf{ch}_k, \mathsf{aux}_k)$ where $\tilde{\mathsf{re}}_k \leftarrow \mathsf{RIP.FwdIss}(\mathsf{re}_k)$ for $k = 0, 1$ and sends $\mathcal{A}$ the tokens σ_b, σ_{1-b}.
- **Output phase.** The adversary $\mathcal{A}$ outputs a guess b'.

We say that a RIP scheme has client privacy if for all λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{CP}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{CP}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpCP}^{\mathcal{A}}_{\mathsf{RIP}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ with the experiment ExpCP defined in Figure 9.

4.2.3 Origin Privacy. We will now define origin privacy for the RIP protocol. Informally, we model the case where the issuer is the only trusted party, while the mediator and the origin are considered to be malicious. We also let all the client's keys, except for the targeted client, be exposed to the adversary. We want to ensure that a collaboration between a mediator and any origin cannot link a valid token with its corresponding RIP issuance execution. Again, any metadata exchanged is outside of the scope of our definition.

Definition 4.5 (Origin Privacy). For RIP and a stateful, polynomial adversary $\mathcal{A}$, we define the experiment $\mathsf{ExpOP}^{\mathcal{A}}_{\mathsf{RIP}}$ as follows:

- **Setup phase.** The adversary $\mathcal{A}$ begins by outputting a pair of public and secret keys for each origin $(\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}^{n_O}_{i=1}) \leftarrow \mathcal{A}(1^\lambda, n_O)$. Upon receiving those key pairs from the adversary, the challenger generates the public and secret key pairs of the issuer and a period secret id for each origin O^j by running $(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{id}}\}^{n_O}_{i=1}) \leftarrow \mathsf{RIP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}\}^{n_O}_{i=1})$. Finally, the challenger generates the key pairs for n_C clients by running $(\{C^i_{\mathsf{sk}}\, C^i_{\mathsf{pk}}\}^{n_C}_{i=1}) \leftarrow \mathsf{RIP.CSetup}(1^\lambda, n_C)$.
- **Query phase.** In this phase, given the public keys of all clients, the adversary $\mathcal{A}$ chooses a target client C^{i^*}. The adversary is then allowed to access the oracles Rq, Iss, Rdm, Update for any client except for the target client C^{i^*}.
- **Challenge phase.** The adversary $\mathcal{A}$ chooses two origins and generates two challenges corresponding to each of the chosen origins, denoted as $\{\mathsf{ch}_k, O^{j_k}_{\mathsf{pk}}\}_{k\in\{0,1\}}$. The challenger in the experiment chooses a random bit b and executes the RIP protocol for the client C^{i^*} and both origins O^{j^*}. The random bit b determines the order of interaction:
  - The challenger first generates a RIP request by executing the request algorithm $(\mathsf{rq}_k, \mathsf{aux}_k) \leftarrow \mathsf{RIP.Rq}(C^{i^*}_{\mathsf{sk}}, O^{j_k}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch}_k)$.
  - The adversary $\mathcal{A}$, on receiving the request rq_k, sends a forward request rq'_k to the challenger.
  - The challenger, on receiving the forward request rq'_k from the adversary, generates a RIP response re by executing the issuing algorithm $(\mathsf{id}_k, \mathsf{re}_k, f) \leftarrow \mathsf{RIP.Iss}(\mathcal{I}_{\mathsf{sk}}, O^{j_k}_{\mathsf{sk}}, O^{k}_{\mathsf{id}}, \mathsf{rq}'_k)$.
  - The challenger then verifies the policy $f(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$. If the request rq passes the policy check, the challenger sends the tuple $(\mathsf{id}_k, \mathsf{re}_k, f)$ to the adversary $\mathcal{A}$. Otherwise, the experiment is aborted.
  - On receiving the response from the challenger, the adversary forwards the response re_k back.
  - The challenger then finalizes a token for the RIP protocol by running $\{\sigma_k, \perp\} \leftarrow \mathsf{RIP.Rdm}(O^{j_k}_{\mathsf{pk}}, \mathsf{re}_k, \mathsf{ch}_k, \mathsf{aux}_k)$.

  After both RIP executions, the challenger sends both tokens (σ_0, σ_1) to the adversary $\mathcal{A}$ if none of the RIP.Rdm executions above fail and $(\perp, \perp)$ otherwise.
- **Output phase.** The adversary $\mathcal{A}$ outputs a guess b'.

We say that a RIP protocol is origin private if for all λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OP}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

$\underline{\mathsf{ExpCP}^{\mathcal{A}}_{\mathsf{RlP}}(1^\lambda, n_C, n_O)}$

$\mathcal{I}_{\mathsf{pk}} \leftarrow \mathcal{A}(1^\lambda)$

$(\{C^i_{\mathsf{pk}}, C^i_{\mathsf{sk}}\}_{i=1}^{n_C}) \leftarrow \mathsf{RlP.CSetup}(1^\lambda, n_C)$

$(\{O^j_{\mathsf{pk}}\}_{i=1}^{n_O}) \leftarrow \mathcal{A}(1^\lambda, n_O)$

$(\{\mathsf{ch}_k, C^{i_k}_{\mathsf{pk}}\}_{k\in\{0,1\}}, O^{j^*}_{\mathsf{pk}}) \leftarrow \mathcal{A}^{\mathsf{FwdIss}}(\{C^i_{\mathsf{sk}}\}_{i=1}^{n_C})$

$b \xleftarrow{\$} \{0,1\}$

for $k = b$ **to** $1-b$ **do**

  $(\mathsf{rq}_k, \mathsf{aux}_k) \leftarrow \mathsf{RlP.Rq}(C^{i_k}_{\mathsf{sk}}, O^{j^*}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch}_k)$

  $\mathsf{rq}'_k \leftarrow \mathsf{RlP.FwdRq}(\mathsf{rq}_k)$

  $(\mathsf{id}_k, \mathsf{re}_k, f) \leftarrow \mathcal{A}(\mathsf{rq}'_k)$

  $\mathsf{re}_k \leftarrow \mathsf{RlP.FwdIss}(\mathsf{id}_k, \mathsf{re}_k, \mathsf{rq}_k, \mathsf{Db}, f)$

  $\sigma_k \leftarrow \mathsf{RlP.Rdm}(O^{j^*}_{\mathsf{pk}}, \mathsf{re}_k, \mathsf{ch}_k, \mathsf{aux}_k)$

if $\prod_{k=0}^{1} \mathsf{RlP.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}_k, \sigma_k) = 1$ **then** : $b' \leftarrow \mathcal{A}(\sigma_0, \sigma_1)$

else $b' \leftarrow \mathcal{A}(\perp, \perp)$

return $b = b'$

$\underline{\mathsf{ExpOP}^{\mathcal{A}}_{\mathsf{RlP}}(1^\lambda, n_C, n_O)}$

$(\{C^i_{\mathsf{pk}}, C^i_{\mathsf{sk}}\}_{i=1}^{n_C}) \leftarrow \mathsf{RlP.CSetup}(1^\lambda, n_C)$

$(\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}_{i=1}^{n_O}) \leftarrow \mathcal{A}(1^\lambda, n_O)$

$(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{id}}\}_{i=1}^{n_O}) \leftarrow \mathsf{RlP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}\}_{i=1}^{n_O})$

$(C^{i^*}_{\mathsf{pk}}, \mathsf{Db}) \leftarrow \mathcal{A}(\{C^i_{\mathsf{pk}}\}_{i=1}^{n_C}, \{O^j_{\mathsf{sk}}\}_{i=1}^{n_O}, \mathcal{I}_{\mathsf{pk}})$

$(\{\mathsf{ch}_k, O^{j_k}_{\mathsf{pk}}\}_{k\in\{0,1\}}, \mathsf{Db}) \leftarrow \mathcal{A}^{\mathsf{Rq,Iss,Rdm,Update}}(\{C^i_{\mathsf{sk}}\}_{i\neq i^*}, \mathsf{Db})$

$b \xleftarrow{\$} \{0,1\}$

for $k = b$ **to** $1-b$ **do**

  $(\mathsf{rq}_k, \mathsf{aux}_k) \leftarrow \mathsf{RlP.Rq}(C^{i^*}_{\mathsf{sk}}, O^k_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch}_k)$

  $\mathsf{rq}'_k \leftarrow \mathcal{A}(\mathsf{rq}_k, \mathsf{Db})$

  $(\mathsf{id}_k, \mathsf{re}_k, f) \leftarrow \mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^{j_l}_{\mathsf{sk}}, O^{j_l}_{\mathsf{id}}\}_{l\in\{0,1\}}, \mathsf{rq}'_k)$

  if $f(\mathsf{id}_k, \mathsf{rq}_k, \mathsf{Db}) = 0$ **then** : **Abort**

  $\mathsf{re}'_k \leftarrow \mathcal{A}(\mathsf{id}_k, \mathsf{re}_k, \mathsf{Db}, f)$

  $\sigma_k \leftarrow \mathsf{RlP.Rdm}(O^{j_k}_{\mathsf{pk}}, \mathsf{re}'_k, \mathsf{ch}_k, \mathsf{aux}_k)$

if $\prod_{k=0}^{1} \mathsf{RlP.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}_k, \sigma_k) = 1$ **then** : $b' \leftarrow \mathcal{A}(\sigma_0, \sigma_1, \mathsf{Db})$

else $b' \leftarrow \mathcal{A}(\perp, \perp, \mathsf{Db})$

return $b = b'$

$\underline{\mathsf{Rq}(i, k, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})}$

if $i = i^*$ **and** $k \in \{j_k\}_{k\in\{0,1\}}$ **then** : **return** $\perp$

$\mathsf{rq} \leftarrow \mathsf{RlP.Rq}(C^i_{\mathsf{sk}}, O^k_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})$

return $(\mathsf{rq}, \mathsf{aux})$

$\underline{\mathsf{Iss}(\mathsf{rq}')}$

$(\mathsf{id}, \mathsf{re}, f) \leftarrow \mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}_{j=1}^{n_O}, \mathsf{rq}')$

return re

$\underline{\mathsf{FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)}$

$\mathsf{re} \leftarrow \mathsf{RlP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$

if $\mathsf{re} \neq \perp$ **then** : $\mathsf{RlP.Update}(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$

return re

$\underline{\mathsf{Rdm}(k, \mathsf{re}, \mathsf{ch}, \mathsf{aux})}$

$\sigma \leftarrow \mathsf{RlP.Rdm}(O^k_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux})$

return σ

$\underline{\mathsf{Update}(\mathsf{id}, \mathsf{rq}, \mathsf{Db})}$

$\mathsf{Db} \leftarrow \mathsf{RlP.Update}(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$

return Db

Figure 9: Experiments for Client and Origin Privacy.

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OP}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpOP}^{\mathcal{A}}_{\mathsf{RlP}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ with the experiment ExpOP defined in Figure 9.

5 GENERIC CONSTRUCTION

This section provides our generic construction from the previously defined cryptographic primitives. Later we analyze the construction's unforgeability, client privacy, and origin privacy properties.

5.1 RlP Generic Construction

Below we provide a formal definition of our generic construction. We also give a high-level overview of the particular algorithms in Figure 10.

Definition 5.1. A *generic rate-limited Privacy Pass* uses a key encapsulation mechanism scheme KEM = (KEM.KGen, KEM.Enc, KEM.Dec), an authenticated encryption with associated data scheme AEAD = (AEAD.Seal, AEAD.Open), a blind signature scheme BS = (BS.KGen, BS.Blind, BS.PreSig, BS.Ext, BS.Vf), a key blinding signature scheme KB = (KB.KGen, KB.Sign, KB.Vf, KB.PKBlind, KB.SKBlind, KB.BlSign, KB.Unblind), a key derivation function KDF and a collision resistance hash function H as building blocks.

$\mathsf{RlP.CSetup}(1^\lambda, n_C)$: The client setup algorithm executes the key generation algorithm $(C^i_{\mathsf{pk}}, C^i_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{KB.KGen}(1^\lambda)$ for each of the n_C clients.

$\mathsf{RlP.OSetup}(1^\lambda, n_O)$: The origin setup algorithm executes the key generation algorithm $(O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{BS.KGen}(1^\lambda)$ for each of the n_O origins.

$\mathsf{RlP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}_{i=1}^{n_O})$: The issuing authority setup algorithm executes the key generation algorithm $(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{KEM.KGen}(1^\lambda)$ for the issuer. On the other hand, with a list of authorized origins, it randomly generates a session id for each of the n_O origins.

$\mathsf{RIP.Rq}(C^i_{\mathsf{sk}}, O^j_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})$: The token request algorithm randomly generates a blinding key bk and a secret seed s. It then hashes the challenge by $\mathsf{ch}' = \mathsf{H}(\mathsf{ch})$, computes $(\tilde{\mathsf{ch}}, \mathsf{aux}) \leftarrow \mathsf{BS.Blind}(O^j_{\mathsf{pk}}, \mathsf{ch}')$ and computes a blinded public key $\mathsf{pk} \leftarrow \mathsf{KB.SKBlind}(C^i_{\mathsf{sk}}, \mathsf{bk})$. The algorithm packs the secret seed, the origin address, and the blinded challenge as $\mathsf{msg}_1 := (s, O^j_{\mathsf{pk}}, \tilde{\mathsf{ch}}, \mathsf{pk})$ and then encrypts $\mathsf{ct} \leftarrow \mathsf{KEM.Enc}(\mathcal{I}_{\mathsf{pk}}, \mathsf{msg}_1)$. It later packs the received ciphertext and the blind public key as $\mathsf{msg}_2 := (\mathsf{pk}, \mathsf{ct})$ and signs $\sigma \leftarrow \mathsf{KB.BlSign}(C^i_{\mathsf{sk}}, \mathsf{bk}, \mathsf{msg}_2)$. Finally, it sets a request as $\mathsf{rq} := (\sigma, \mathsf{msg}_2, \mathsf{bk})$.

$\mathsf{RIP.FwdRq}(C^i_{\mathsf{pk}}, \mathsf{rq})$: The proxy request algorithm extracts the information from the request $(\sigma, \mathsf{msg}, \mathsf{bk}) := \mathsf{rq}$, unblinds the public key $\mathsf{pk}' \leftarrow \mathsf{KB.Unblind}(\mathsf{pk}, \mathsf{bk})$ and verifies the signature $\mathsf{KB.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$. The algorithm aborts if either the blinded public key does not originate from the requesting client or the signature is invalid. Otherwise, it removes the blind factor bk and outputs the new request rq'.

$\mathsf{RIP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}^{n_O}_{j=1}, \mathsf{rq}')$: The issue algorithm extracts the information from the forwarded request by $(\sigma, \mathsf{msg}) := \mathsf{rq}'$ and $(\mathsf{pk}, \mathsf{ct}) := \mathsf{msg}$. It first verifies the validity of the signature σ and aborts if the signature is invalid. The algorithm then decrypts the ciphertext by $(s, O_{\mathsf{pk}}, \tilde{\mathsf{ch}}, \mathsf{pk}) \leftarrow \mathsf{KEM.Dec}(\mathcal{I}_{\mathsf{sk}}, \mathsf{ct})$. It then verifies the consistency of the public key pk in the request rq and the public key received from the ciphertext ct and aborts if these two keys are different. Upon receiving the plaintext, it computes the pre-signature for the blinded challenge $\sigma' \leftarrow \mathsf{BS.PreSig}(O^j_{\mathsf{sk}}, \tilde{\mathsf{ch}})$ and derives a blinded session id by $id \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, O^j_{\mathsf{id}})$. It then derives a key K by generating a random salt, computes $K \leftarrow \mathsf{KDF}(s, \mathsf{salt})$ and seals the pre-signature $\mathsf{ct}' \leftarrow \mathsf{AEAD.Seal}(K, \sigma')$. Finally, the algorithm sets the response as $\mathsf{re} := (\mathsf{ct}', \mathsf{salt})$ and outputs the tuple $(\mathsf{id}, \mathsf{re}, f)$ where f is the policy corresponding to the provided origin.

$\mathsf{RIP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)$: The policy enforcement algorithm, on the input of the database Db, enforces the policy by executing $f(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$. If it passes the policy check, the algorithm forwards the response re to the requesting client. Otherwise, it aborts the protocol.

$\mathsf{RIP.Rdm}(O^j_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux})$: The token redeem algorithm extracts the information from the response $(\mathsf{ct}, \mathsf{salt}) := \mathsf{re}$ and derives the key $K \leftarrow \mathsf{KDF}(s, \mathsf{salt})$. It then open the ciphertext $\sigma' \leftarrow \mathsf{AEAD.Open}(K, \mathsf{re})$ and finalizes the token by computing $\sigma \leftarrow \mathsf{BS.Ext}(O^j_{\mathsf{pk}}, \sigma', \mathsf{aux})$.

$\mathsf{RIP.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}, \sigma)$: The verification algorithm verifies the validity of the token by executing $\mathsf{BS.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}, \sigma)$.

$\mathsf{RIP.Update}(\mathsf{id}, \mathsf{Db})$: The update subprocedure extracts the information from the request $(\sigma, \mathsf{msg}, \mathsf{bk}) := \mathsf{rq}$ and database $(\{\mathsf{id}^j, \mathsf{info}^j\}^n_{j=1}) := \mathsf{Db}$. It then derives the per-user-origin period id by computing $\mathsf{id}' \leftarrow \mathsf{KB.Unblind}(\mathsf{id}, \mathsf{bk})$. It looks up id' in the database and updates its value based on the structure of Db. If id' is not in the database, the algorithm includes id' in the database Db and initiates its value.

$f(\mathsf{id}, \mathsf{rq}, \mathsf{Db})$: The policy enforcement function extracts the information from the request $(\sigma, \mathsf{msg}, \mathsf{bk}) := \mathsf{rq}$ and database $(\{\mathsf{id}^j, \mathsf{info}^j\}^n_{j=1}) := \mathsf{Db}$. It then derives the per-user-origin period id by computing $\mathsf{id}' \leftarrow \mathsf{KB.Unblind}(\mathsf{id}, \mathsf{bk})$. Together with the database, the algorithm outputs 1 if id' passes the policy and 0 otherwise.

For applications that do not require maintaining a database (e.g., geolocation policy), we exclude the update algorithm RIP.Update and the database Db from the construction.

5.2 Security Analysis

In this section, we will analyze the security of the RIP construction presented in Definition 5.1. We will begin with one-more unforgeability, followed by client and origin privacy.

One-more Unforgeability.

Theorem 5.2. *For all λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OMUF}} \leq \mathbf{Adv}^{H,\mathcal{A}_1}_{\mathsf{CR}} + n_C \cdot \mathbf{Adv}^{\mathcal{A}_2}_{\mathsf{KBlind}} + \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{KB.UF}} + \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{BS.OMUF}}.$$

Proof. We show the statement via a sequence of games.

Game G_0: This is the real game where the client tries to breach the policy of an origin. We have: $\mathbf{Adv}_0 = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{OMUF}}$.

Game G_1: This game is like G_0, but we let the game abort whenever there is a collision in the hash function H. Let $\mathcal{A}_1$ be an adversary that produces two challenges ch_1 and ch_2 such that $\mathsf{H}(\mathsf{ch}_1) = \mathsf{H}(\mathsf{ch}_2)$, then we have: $|\mathbf{Adv}_0 - \mathbf{Adv}_1| \leq \mathbf{Adv}^{\mathsf{H},\mathcal{A}}_{\mathsf{CR}}$.

Game G_2: This game is like G_0, but we abort whenever the mediator derives a fresh id'. Since the key blinding signature scheme has the double key blinding property, a malicious client should have no advantage. We have: $\mathbf{Adv}_1 = \mathbf{Adv}_2$.

Game G_3: This game is like G_2, but we let the public key pk_2 of the client C^2 to be generated by $\mathsf{pk}_2 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}_1, \mathsf{bk}_2)$ where bk_2 is chosen by the challenger. Clearly, game G_3 is indistinguishable from game G_2 except for an adversary $\mathcal{A}_2$ that breaks the ExpKBlind experiment. Next, let us do the same for the other $n_C - 2$ clients. Through a series of hybrid arguments, we have: $|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq n_C \cdot \mathbf{Adv}^{\mathcal{A}_2}_{\mathsf{KBlind}}$.

Game G_4: This game is like G_3, except that we let the game abort whenever $\mathcal{A}$ queries for a request $\mathsf{rq} = (\sigma, \mathsf{msg}, \mathsf{bk})$ with $\mathsf{msg} = (\mathsf{pk}, \mathsf{ct})$ and $\mathsf{KB.Unblind}(\mathsf{pk}, \mathsf{bk}) = C^j_{\mathsf{pk}}$ for some $j \neq i^*$. We show that the advantage of an adversary against the unforgeability game of the key blind signature bounds the probability gap of this game and G_3.

Let $\mathcal{A}$ be a distinguisher between game G_4 and G_3. We then construct the adversary $\mathcal{A}_3$ that runs $\mathcal{A}$ as a sub-algorithm to break the unforgeability of a key blinding signature scheme.

- $\mathcal{A}_3$ receives a challenge public key pk.
- Let $C^1_{\mathsf{pk}} \leftarrow \mathsf{pk}$ and generate the other $n_C - 1$ client public keys as above to construct the list of n_C authenticate client

Setup:

$\underline{\mathsf{RlP.CSetup}(1^\lambda, n_C)}$

for $i = 1$ **to** n_C **do**

$\quad (C^i_{\mathsf{pk}}, C^i_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{KB.KGen}(1^\lambda)$

return $\{C^i_{\mathsf{pk}}, C^i_{\mathsf{sk}}\}_{i=1}^{n_C}$

$\underline{\mathsf{RlP.OSetup}(1^\lambda, n_O)}$

for $j = 1$ **to** n_O **do**

$\quad (O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{BS.KGen}(1^\lambda)$

return $\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}_{j=1}^{n_O}$

$\underline{\mathsf{RlP.ISetup}(1^\lambda, \{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}\}_{i=1}^{n_O})}$

$(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}) \xleftarrow{\$} \mathsf{KEM.KGen}(1^\lambda)$

for $j = 1$ **to** n_O **do**

$\quad O^j_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$

return $(\mathcal{I}_{\mathsf{pk}}, \mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{id}}\}_{j=1}^{n_O})$

Main Part:

$\underline{\mathsf{RlP.Rq}(C^i_{\mathsf{sk}}, O^j_{\mathsf{pk}}, \mathcal{I}_{\mathsf{pk}}, \mathsf{ch})}$

$(\mathsf{bk}, s) \xleftarrow{\$} \{0,1\}^\lambda$

$\mathsf{ch}' \leftarrow \mathsf{H}(\mathsf{ch})$

$(\tilde{\mathsf{ch}}, \mathsf{aux}) \leftarrow \mathsf{BS.Blind}(O^j_{\mathsf{pk}}, \mathsf{ch}')$

$\mathsf{pk} \leftarrow \mathsf{KB.SKBlind}(C^i_{\mathsf{sk}}, \mathsf{bk})$

$\mathsf{msg}_1 := (s, O^j_{\mathsf{pk}}, \tilde{\mathsf{ch}}, \mathsf{pk});\ \mathsf{ct} \leftarrow \mathsf{KEM.Enc}(\mathcal{I}_{\mathsf{pk}}, \mathsf{msg}_1)$

$\mathsf{msg}_2 := (\mathsf{pk}, \mathsf{ct});\ \sigma \leftarrow \mathsf{KB.BlSign}(C^i_{\mathsf{sk}}, \mathsf{bk}, \mathsf{msg}_2)$

$\mathsf{rq} := (\sigma, \mathsf{msg}_2, \mathsf{bk})$

return $(\mathsf{rq}, \mathsf{aux})$

$\underline{\mathsf{RlP.Iss}(\mathcal{I}_{\mathsf{sk}}, \{O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}_{j=1}^{n_O}, \mathsf{rq}')}$

Parse $(\sigma, \mathsf{msg}) := \mathsf{rq}'$

Parse $(\mathsf{pk}, \mathsf{ct}) := \mathsf{msg}$

if $\mathsf{KB.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma) = 0$ **then** : **Abort**

$(s, O_{\mathsf{pk}}, \tilde{\mathsf{ch}}, \mathsf{pk}') \leftarrow \mathsf{KEM.Dec}(\mathcal{I}_{\mathsf{sk}}, \mathsf{ct})$

if $\mathsf{pk} \neq \mathsf{pk}'$ **then** : **Abort**

for $j = 1$ **to** n_O **do**

$\quad$ **if** $O_{\mathsf{pk}} = O^j_{\mathsf{pk}}$ **then** :

$\quad\quad \sigma' \leftarrow \mathsf{BS.PreSig}(O^j_{\mathsf{sk}}, \tilde{\mathsf{ch}})$

$\quad\quad \mathsf{id} \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, O^j_{\mathsf{id}})$

$\quad\quad \mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda;\ K \leftarrow \mathsf{KDF}(s, \mathsf{salt})$

$\quad\quad \mathsf{ct}' \leftarrow \mathsf{AEAD.Seal}(K, \sigma')$

$\quad\quad \mathsf{re} := (\mathsf{ct}', \mathsf{salt})$

return $(\mathsf{id}, \mathsf{re})$

$\underline{\mathsf{RlP.Rdm}(O^j_{\mathsf{pk}}, \mathsf{re}, \mathsf{ch}, \mathsf{aux})}$

Parse $(\mathsf{ct}, \mathsf{salt}) := \mathsf{re}$

$K \leftarrow \mathsf{KDF}(s, \mathsf{salt})$

$\sigma' \leftarrow \mathsf{AEAD.Open}(K, \mathsf{re})$

$\sigma \leftarrow \mathsf{BS.Ext}(O^j_{\mathsf{pk}}, \sigma', \mathsf{aux})$

return σ

$\underline{\mathsf{RlP.Vf}(O^j_{\mathsf{pk}}, \mathsf{ch}, \sigma)}$

$b \leftarrow \mathsf{BS.Vf}(O^j_{\mathsf{pk}}, \mathsf{H}(\mathsf{ch}), \sigma)$

return b

$\underline{\mathsf{RlP.Update}(\mathsf{id}, \mathsf{rq}, \mathsf{Db})}$

Parse $(\{\mathsf{id}^j, \mathsf{info}^j\}_{j=1}^n) := \mathsf{Db}$

Parse $(\sigma, \mathsf{msg}, \mathsf{bk}) := \mathsf{rq}$

$\mathsf{id}' \leftarrow \mathsf{KB.Unblind}(\mathsf{id}, \mathsf{bk})$

for $j = 1$ **to** n **do**

$\quad$ **if** $\mathsf{id}' = \mathsf{id}^j$ **then** :

$\quad\quad \mathsf{info}^j \leftarrow \mathsf{Update}(\mathsf{info}^j)$

if $\mathsf{id}' \notin \mathsf{Db}$ **then** :

$\quad \mathsf{Db} = \mathsf{Db} \cup \{\mathsf{id}', \mathsf{Update}(0)\}$

return Db

$\underline{\mathsf{RlP.FwdRq}(C^i_{\mathsf{pk}}, \mathsf{rq})}$

Parse $(\sigma, \mathsf{msg}, \mathsf{bk}) := \mathsf{rq}$

$\mathsf{pk}' \leftarrow \mathsf{KB.Unblind}(\mathsf{pk}, \mathsf{bk})$

if $C^i_{\mathsf{pk}} \neq \mathsf{pk}'$ **or** $\mathsf{KB.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma) = 0$ **then** : **Abort**

$\mathsf{rq}' := (\sigma, \mathsf{msg})$

return rq'

$\underline{\mathsf{RlP.FwdIss}(\mathsf{id}, \mathsf{re}, \mathsf{rq}, \mathsf{Db}, f)}$

if $f(\mathsf{id}, \mathsf{rq}, \mathsf{Db}) = 1$ **then** : **return** re

else return $\perp$

Figure 10: A Generic Construction for Private Access Token Schemes.

public keys for $\mathcal{A}$. The adversary $\mathcal{A}_3$ keeps a list of the public key and its corresponding blind key.

- $\mathcal{S}$ generates an issuer and origin key pair and then sends $\mathcal{A}$ the corresponding public keys.
- When $\mathcal{A}$ starts a RlP execution for client C^i, the adversary $\mathcal{A}_3$ queries BlSign and simulates the execution using the corresponding blind key.
- The adversary $\mathcal{A}_3$ uses the response from $\mathcal{A}$ together with the blind key bk_i that corresponds to $\mathcal{A}$'s target client C^i to derive a forgery for the KB scheme's challenger.

Thus, we have: $|\mathbf{Adv}_4 - \mathbf{Adv}_3| \leq \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{KB.UF}}$.

Finally, we bound the probability $\mathbf{Adv}_4$ that G_4 outputs 1. Given an adversary $\mathcal{A}$ against game G_4, we then construct the adversary $\mathcal{A}_4$ against the one-more unforgeability game of the blind signature.

- In the setup phase of $\mathcal{A}$, the adversary $\mathcal{A}_4$ follows the description of the unforgeability game.
- On receiving $C^{i^*}_{\mathsf{pk}}$ from $\mathcal{A}$, $\mathcal{A}_4$ forwards it to the challenger. $\mathcal{A}_4$ gets access to the oracle BS.PreSig.
- $\mathcal{A}_4$ follows the description on each query of $\mathcal{A}$ to the first oracle (the query for a token challenge). Briefly, in order to answer each query of $\mathcal{A}$, $\mathcal{A}_4$ makes a query to the BS.PreSig oracle
- $\mathcal{A}_4$ uses the output of $\mathcal{A}$ as its output.

$\mathcal{A}_4$ wins whenever $\mathcal{A}$ wins, and we have: $\mathbf{Adv}_4 \leq \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{BS.OMUF}}$. □

Client Privacy.

THEOREM 5.3. *For all λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:*

$$\mathbf{Adv}_{\mathsf{CP}}^{\mathcal{A}} \leq 4n_C \cdot \mathbf{Adv}_{\mathsf{KBlind}}^{\mathcal{A}_1} + \mathbf{Adv}_{\mathsf{Blind}}^{\mathcal{A}_2}.$$

PROOF. We show the statement via a sequence of games.

Game G_0: This is the real client privacy game in Figure 9. By definition, we have $\mathbf{Adv}_0 = \mathbf{Adv}_{\mathsf{CP}}^{\mathcal{A}}$.

Game G_1: This game is like G_0, but we replace the blinded public key and the key-blind signature of client i_b with a random key and its corresponding signature. We will show that G_1 is indistinguishable from G_0 except for an adversary $\mathcal{A}_1$ breaking the key blinding game of the signature. Assuming that a PPT adversary $\mathcal{A}$ can tell apart G_0 and G_1, we construct $\mathcal{A}_1$ as follows.

- $\mathcal{A}_1$ receives from the challenger a key pair $(\mathsf{sk}^*, \mathsf{pk}^*)$ and another public key pk_β, where β is a random bit, $\mathsf{pk}_0 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}, \mathsf{bk})$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{KB.KGen}(1^\lambda)$.
- It randomizes $i^* \xleftarrow{\$} [n_C]$, sets $\left(C_{\mathsf{sk}}^{i^*}, C_{\mathsf{pk}}^{i^*}\right) \leftarrow (\mathsf{sk}^*, \mathsf{pk}^*)$ and generates $n_C - 1$ client key pairs to construct the list of n_C client public keys for $\mathcal{A}$.
- After receiving the issuer and origin public keys from $\mathcal{A}$, $\mathcal{A}_1$ perfectly follows the description of gameG_0 on each query of $\mathcal{A}$ to the FwdRq, FwdIss oracles.
- $\mathcal{A}_1$ receives a target origin and two target client public keys. The latter are indexed by i_0 and i_1. It then chooses a random bit b. If $i_b = i^*$, it continues; otherwise, it aborts.
- On receiving $\mathsf{ch}_0, \mathsf{ch}_1$ from $\mathcal{A}$, $\mathcal{A}_1$ follows the description of the challenge phase for $\mathcal{A}$ except for two changes in rq_b: 1) It sets $\mathsf{pk} \leftarrow \mathsf{pk}_\beta$, and 2) It queries its challenger for the signature σ on msg_2.

Finally, $\mathcal{A}_1$ outputs 1 whenever $\mathcal{A}$ wins its own game. It is clear that for each case $\beta = 0$ and $\beta = 1$, $\mathcal{A}_1$ is simulating exactly the games G_0 and G_1 for $\mathcal{A}$, respectively. Let $\mathbf{Adv}_1$ be the advantage of $\mathcal{A}$ in G_1, then we have:

$$\begin{aligned}|\mathbf{Adv}_0 - \mathbf{Adv}_1| &\leq n_C \cdot |\Pr[0 \leftarrow \mathcal{A}_1 \mid \beta = 0] - \Pr[1 \leftarrow \mathcal{A}_1 \mid \beta = 1]| \\ &\leq 2n_C \cdot \mathbf{Adv}_{\mathsf{KBlind}}^{\mathcal{A}_1},\end{aligned}$$

where $\mathbf{Adv}_{\mathsf{KBlind}}^{\mathcal{A}_1}$ is the advantage of $\mathcal{A}_1$ for its own the key-blinding game.

Game G_2: This game is as G_1 except that the blinded public key and the key-blind signature of client i_{1-b} are replaced by a random key and its corresponding signature. With a similar argument as above, we obtain:

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq 2n_C \cdot \mathbf{Adv}_{\mathsf{KBlind}}^{\mathcal{A}_1}.$$

Now, what is left to be shown is an upper bound for the advantage of game G_2. The adversary view in game G_2 is independent of $C_{\mathsf{pk}}^{i_0}$ and $C_{\mathsf{pk}}^{i_1}$, so it is left to link the execution with a challenge-token pair to win the game. Indeed, we will show that if there exists a PPT adversary $\mathcal{A}$ against G_2, then there exists an adversary $\mathcal{A}_2$ against the blindness of the blind signature.

- $\mathcal{A}_2$ honestly generates n_C key pairs $(\{C_{\mathsf{pk}}^i, C_{\mathsf{sk}}^i\}_{i=1}^{n_C})$ for $\mathcal{A}$. On receiving $O_{\mathsf{pk}}^j, \mathcal{I}_{\mathsf{pk}}$ from $\mathcal{A}$, $\mathcal{A}_2$ assigns its own blind signature public key as $\mathsf{pk} \leftarrow O_{\mathsf{pk}}$.
- When receiving $\mathsf{ch}_0, \mathsf{ch}_1$ from $\mathcal{A}$, $\mathcal{A}_2$ assigns $\mathsf{msg}_0 \leftarrow \mathsf{ch}_0, \mathsf{msg}_1 \leftarrow \mathsf{ch}_1$.
- It chooses two random key pairs for the key-blind signature used in the two executions with $\mathcal{A}$. On receiving from the challenger two blinded messages $\mathsf{msg}'_b, \mathsf{msg}'_{1-b}$ in a random order indicated by the unknown bit b, $\mathcal{A}_2$ assigns $\tilde{\mathsf{ch}}_k \leftarrow \mathsf{msg}'_k$ for $k = b, 1-b$. The requests and ciphertexts are computed as in the protocol and then concatenated in the corresponding requests $\tilde{\mathsf{rq}}_b, \tilde{\mathsf{rq}}_{1-b}$.
- After receiving $\mathsf{re}_b, \mathsf{re}_{1-b}$ from $\mathcal{A}$, $\mathcal{A}_2$ decrypts to obtain pre-signatures σ'_b, σ'_{1-b} in order to answer its challenger.
- On receiving redeemed tokens from the challenger, $\mathcal{A}_2$ forwards them to $\mathcal{A}$. It outputs what $\mathcal{A}$ outputs.

It is clear that $\mathcal{A}_2$ wins whenever $\mathcal{A}$ wins. Let $\mathbf{Adv}_2$ be the advantage of $\mathcal{A}$ in game G_2, then we have: $\mathbf{Adv}_2 \leq \mathbf{Adv}_{\mathsf{Blind}}^{\mathcal{A}_2}$ and $\mathbf{Adv}_0 \leq \mathbf{Adv}_1 + \mathbf{Adv}_2$ as a consequence. □

Origin Privacy.

THEOREM 5.4. *For all λ, for any polynomial time algorithm $\mathcal{A}$ with q queries to the key derivation function KDF, the following holds:*

$$\begin{aligned}\mathbf{Adv}_{\mathsf{OP}}^{\mathcal{A}} \leq \mathbf{Adv}_{(q,q)\text{-}\mathsf{PRF}}^{KDF,\mathcal{A}_1} + 2 \cdot (&\mathbf{Adv}_{\mathsf{AEAD}}^{\mathcal{A}_3} + \mathbf{Adv}_{\mathsf{KEM}}^{\mathcal{A}_2} \\ &+ n_C \cdot n_O \cdot \mathbf{Adv}_{\mathsf{BrKBlind}}^{\mathcal{A}_4}).\end{aligned}$$

PROOF. We show the statement via a sequence of games.

Game G_0: This is the real origin privacy experiment described in Figure 9. By definition, we have: $\mathbf{Adv}_0 = \mathbf{Adv}_{\mathsf{OP}}^{\mathcal{A}}$.

Game G_1: This game is like G_0 but aborts when the adversary $\mathcal{A}$ queries the KDF with the valid pair (s, salt). As the key of the KDF is sampled uniformly from $\{0,1\}^\lambda$ while salt is publicly transmitted, we have: $|\mathbf{Adv}_1 - \mathbf{Adv}_0| \leq \mathbf{Adv}_{(q,q)\text{-}\mathsf{PRF}}^{\mathsf{KDF},\mathcal{A}_1}$.

Game G_2: This game is like G_1, but we replace $(K_b, O_{\mathsf{pk}}^{j_b}, \mathsf{ch}_b, \mathsf{pk}_b)$ with a random triplet $(K^*, O_{\mathsf{pk}}^*, \mathsf{ch}^*, \mathsf{pk}^*)$. We will show that G_2 is indistinguishable from G_1 except for an adversary $\mathcal{A}_2$ breaking the $\mathsf{IND\text{-}CCA}$ experiment of the KEM scheme related to $\mathcal{I}_{\mathsf{pk}}$. Let $\mathcal{A}$ be an algorithm distinguishing between games G_2 and G_1. We then construct the adversary $\mathcal{A}_2$ that uses $\mathcal{A}$ as a subprotocol. The algorithm $\mathcal{A}_2$ first receives two challenge-origin pairs $\{\mathsf{ch}_i, O_{\mathsf{sk}}^i\}$ from $\mathcal{A}$. Upon receiving the challenge-origin pairs, $\mathcal{A}_2$ randomly generates a key K_b and sends $(K_b, O_{\mathsf{pk}}^{j_b}, \mathsf{ch}_b, \mathsf{pk}_b)$ along with a random tuple $(K^*, O_{\mathsf{pk}}^*, \mathsf{ch}^*, \mathsf{pk}^*)$ to the challenger of the $\mathsf{IND\text{-}CCA}$ experiment. The challenger then randomly chooses one of the two received messages, encrypts it with the key $\mathcal{I}_{\mathsf{pk}}$, and sends back the corresponding ciphertext ct to $\mathcal{A}_2$. The algorithm $\mathcal{A}_2$ then forwards the received ciphertext ct to $\mathcal{A}$ and outputs the answer of the distinguisher $\mathcal{A}$. It is clear that $\mathcal{A}_2$ breaks the $\mathsf{IND\text{-}CCA}$ experiment whenever $\mathcal{A}$ successfully distinguishes between games G_2 and G_1. Hence, let $\mathbf{Adv}_2$ be the advantage of $\mathcal{A}$ in the game G_2, then we have:

$|\mathbf{Adv}_2 - \mathbf{Adv}_1| \leq \mathbf{Adv}^{\mathcal{A}_2}_{\mathsf{KEM}}$.

Game G_3: This game is like G_2, but we replace $(K_{1-b}, O^{j_{1-b}}_{\mathsf{pk}}, \mathsf{ch}_{1-b}, \mathsf{pk}_{1-b})$ with a random tuple $(K^*, O^*_{\mathsf{pk}}, \mathsf{ch}^*, \mathsf{pk}^*)$. Let $\mathbf{Adv}_3$ be the advantage of $\mathcal{A}$ in G_3, using the same arguments as above we have: $|\mathbf{Adv}_3 - \mathbf{Adv}_2| \leq \mathbf{Adv}^{\mathcal{A}_2}_{\mathsf{KEM}}$.

Game G_4: This game is like G_3, but we replace the ciphertext in the response re_b with the encryption of a random message msg. We will show that G_4 is indistinguishable from G_3 except for an adversary $\mathcal{A}_3$ breaking the IND-CCA experiment of the AEAD scheme. Let $\mathcal{A}$ be an algorithm distinguishing between games G_4 and G_3. We then construct an algorithm $\mathcal{A}_3$ that uses $\mathcal{A}$ as a subprotocol. $\mathcal{A}_3$ first receive two challenge-origin pairs $\{\mathsf{ch}_k, O^{j_k}_{\mathsf{sk}}\}$ from $\mathcal{A}$. Upon receiving the challenge-origin pairs, $\mathcal{A}_3$ generates the corresponding pre-signature σ'_b and sends the pre-signature and the random message msg to the challenger of the IND-CCA experiment. The challenger then randomly chooses one of the two received messages and sends back the corresponding ciphertext ct. The algorithm $\mathcal{A}_3$ then forwards the received ciphertext ct to $\mathcal{A}$ and outputs the answer of the distinguisher $\mathcal{A}$. It is clear that $\mathcal{A}_3$ breaks the IND-CCA experiment whenever $\mathcal{A}$ successfully distinguishes between games G_4 and G_3. Hence, let $\mathbf{Adv}_3$ be the advantage of $\mathcal{A}$ in the game G_3, then we have: $|\mathbf{Adv}_4 - \mathbf{Adv}_3| \leq \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{AEAD}}$.

Game G_5: This game is like G_4, but we replace the ciphertext in re_{1-b} with the encryption of a random message. Similar to the argument in G_4, we have: $|\mathbf{Adv}_5 - \mathbf{Adv}_4| \leq \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{AEAD}}$.

Game G_6: This game is like G_5, but we replace the index id_b with a random index id. We will show that G_6 is indistinguishable from G_5 except for an algorithm $\mathcal{A}_4$ breaking the ExpBrKBlind experiment of the KB scheme. Let $\mathcal{A}$ be an algorithm distinguishing between game G_6 and G_5. We then construct the algorithm $\mathcal{A}_4$ that uses $\mathcal{A}$ as a subprotocol. The adversary $\mathcal{A}_4$ first receives a pair of public keys $(\mathsf{pk}^*, \mathsf{pk}_{b'})$ with $b' \in \{0,1\}$ from the challenger in the ExpBrKBlind experiment where $\mathsf{pk}_0 \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}^*, \mathsf{bk}^*)$, $\mathsf{pk}_1 \xleftarrow{\$} \mathsf{KB.KGen}(1^\lambda)$ and b is a random bit. Upon receiving the challenge, the algorithm $\mathcal{A}_4$ randomly chooses one position $j_b \xleftarrow{\$} [n_O]$ and generates a pair of public and secret keys $(O^{j_b}_{\mathsf{pk}}, O^{j_b}_{\mathsf{sk}})$ while honestly generating the triplet $\{O^j_{\mathsf{pk}}, O^j_{\mathsf{sk}}, O^j_{\mathsf{id}}\}$ for other positions. $\mathcal{A}_4$ also randomly chooses one position $i^* \xleftarrow{\$} [n_C]$ and embeds pk^* as the public key $C^{i^*}_{\mathsf{pk}}$. As a result, this game aborts when $\mathcal{A}$ chooses a different target client from the client C^{i^*}. To simulate the view for $\mathcal{A}$, the adversary $\mathcal{A}_4$ then executes as follows:

- For the request oracle Rq , if $\mathcal{A}$ makes a query for the target client i^*, $\mathcal{A}_4$ simulates a ciphertext ct, randomly generates a blind key bk, computes the blinded public key $\mathsf{pk}' \leftarrow \mathsf{KB.PKBlind}(\mathsf{pk}^*, \mathsf{bk})$ and makes a query to the $\mathsf{BlSign}(\mathsf{bk}, \mathsf{msg})$ oracle in the ExpBrKBlind experiment where $\mathsf{msg} := (\mathsf{pk}', \mathsf{ct})$. $\mathcal{A}_4$ honestly executes a request algorithm RlP.Rq otherwise. We use a similar argument when the algorithm $\mathcal{A}_4$ has to execute a request algorithm RlP.Rq in the challenge phase.
- For the issue oracle Iss, if $\mathcal{A}$ makes a query for the target origin j_b and the client i with $i \neq i^*$, the adversary $\mathcal{A}_4$ makes a query to the $\mathsf{KBlind}(\mathsf{sk})$ oracle in the ExpBrKBlind experiment where sk is the secret key of the client C^i together with the blind key bk extracted from the forwarded request rq' to derive the session index id while simulating the corresponding response re for the request. $\mathcal{A}_4$ honestly executes an issue algorithm RlP.Iss otherwise. When the algorithm $\mathcal{A}_4$ executes an issue algorithm RlP.Rq in the challenge phase, $\mathcal{A}_4$ embeds the challenge of the ExpBrKBlind experiment in the session index id of the RlP.Iss execution if it corresponds to the origin O^{j_b} while simulating the response re and honestly executing the RlP.Iss algorithm otherwise.

Finally, $\mathcal{A}_4$ forwards the output of $\mathcal{A}$ to the challenger in ExpBrKBlind. It is clear that $\mathcal{A}_4$ breaks the ExpBrKBlind experiment when $\mathcal{A}$ successfully distinguishes between games G_6 and G_5 with the target origin O^{j_b} and client C^{i^*}. Therefore, with probability $\frac{1}{n_C \cdot n_O}$, $\mathcal{A}_4$ breaks the ExpBrKBlind experiment. Hence, let $\mathbf{Adv}_6$ be the advantage of $\mathcal{A}$ in G_6, then we have: $|\mathbf{Adv}_6 - \mathbf{Adv}_5| \leq n_C \cdot n_O \cdot \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{BrKBlind}}$.

Game G_7: This game is like G_6, but we replace the index id_{1-b} with a random index id. Following the same argument as in game G_6, we have: $|\mathbf{Adv}_7 - \mathbf{Adv}_6| \leq n_C \cdot n_O \cdot \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{BrKBlind}}$.

Now what is left to be shown is an upper bound for the advantage of game G_7. Since we replace all elements that the challenger sends to the adversary in the ExpOP experiments with random ones, the adversary should have no additional advantage in guessing the random bit b. Therefore, the statement follows.

□

6 DISCUSSION

This section briefly discusses the potential post-quantum instantiation of our generic construction and the security of the instantiation proposed in the IETF draft [20].

6.1 Post-quantum Instantiation

The RlP scheme proposed in the IETF draft is not post-quantum secure since it relies on discrete logarithm and RSA-based assumptions. A simple way to build a post-quantum secure RlP scheme would be to instantiate our generic construction with post-quantum building blocks. We only use collision-resistant properties of the hash functions and do not require the random oracle heuristic. Therefore, such an instantiation should work without problems related to using the quantum random oracle model. Unfortunately, no post-quantum secure key-blinding signature scheme currently fulfills our definition; they only support weaker notions [15]. It is worth noting that our definitions of indistinguishability are essential for our RlP construction. We leave constructing such a signature scheme as an open problem.

Fortunately, all other cryptographic building blocks can be instantiated in the post-quantum setting. In the following, we will briefly discuss the potential instantiation. NIST has chosen four quantum-resistant cryptographic algorithms, and CRYSTAL-KYBER [7] was selected as a key encapsulation mechanism [28] and can be used as part of our construction. Instantiating the two-move blind signature can be done with the recently proposed post-quantum two-round blind signature scheme [1]. Finally, to instantiate the key derivation function and authenticate encryption schemes, we can use the standard technique to make symmetric primitives post-quantum secure by increasing the parameters [22].

It is worth noting that independent of our work, Policharla et al. proposed a post-quantum instantiation of Privacy Pass [30]. The idea is to use anonymous credentials as the building block for the token instead of blind signatures, which, as we noticed previously, are single-use and single-attribute anonymous credentials. They show that their construction can also be used to rate-limit users, and thus they show how to construct a post-quantum rate-limited Privacy Pass. We want to highlight here that their construction relies on a tailored anonymous credentials system, inherently different from our generic construction.

6.2 IETF Draft Security Analysis

Let us define an instantiation of our generic construction of RlP that is consistent with the IETF draft [20]. First, we use RSA-KEM [32] as the key encapsulation mechanism. For the blind signature scheme and key-blinding signature scheme, we pick RSA-BSSA [12] and Schnorr signature scheme with key-blinding functionality [11]. We instantiate the Extract and Expand for the key derivation KDF function using HMAC [23, 24]. We also use SHA for the implementation of the hash function H.

Combining theorem 5.2 and theorem 3.5, we derive the unforgeability advantage against this instantiation as follows: $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OMUF}} \leq \mathbf{Adv}^{\mathsf{H},\mathcal{A}_1}_{\mathsf{CR}} + \mathbf{Adv}^{\mathcal{A}_2}_{\mathsf{DL}} + \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{BS.OMUF}}$, where the collision resistance of H immediately follows from the collision resistance of SHA [18]. We omit the term $\mathbf{Adv}_{\mathsf{KBlind}}$ due to the perfect blindness of the key-blinding Schnorr signature scheme. We now proceed with client privacy. From theorem 5.3, blindness of the RSA-BSSA blind signature scheme, together with theorem 3.5, we have: $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{CP}} \leq \mathsf{negl}(\lambda)$. Due to the perfect blindness of key-blinding Schnorr, the term $\mathbf{Adv}_{\mathsf{KBlind}}$ disappears for the same reason. However, this is not the case for the blindness experiment of the RSA-BSSA blind signature scheme, which only holds statistically. This term vanishes in $\mathsf{negl}(\lambda)$. We conclude that the RlP instantiation proposed in the IETF draft [20] achieves statistical client privacy.

We will now analyze the origin privacy property, which can be derived from the composition of theorem 5.4 and theorem 3.5:

$$\begin{aligned}\mathbf{Adv}^{\mathcal{A}}_{\mathsf{OP}} \leq & \mathbf{Adv}^{\mathsf{KDF},\mathcal{A}_1}_{(q,q)\text{-}\mathsf{PRF}} + 2 \cdot (\mathbf{Adv}^{\mathsf{AES},\mathcal{A}_2}_{\mathsf{CCA}} \\ & + \mathbf{Adv}^{\mathcal{A}_3}_{\mathsf{RSA}} + n_C \cdot n_O \cdot \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{DDH}}).\end{aligned}$$

With $\mathbf{Adv}^{\mathsf{KDF},\mathcal{A}_1}_{(q,q)\text{-}\mathsf{PRF}} \leq \mathbf{Adv}^{\mathsf{Extract},\mathcal{B}_1}_{\mathsf{CR}} + \mathbf{Adv}^{\mathsf{Extract},\mathcal{B}_2}_{(q,q)\text{-}\mathsf{PRF}} + \mathbf{Adv}^{\mathsf{Expand},\mathcal{B}_3}_{(q,q)\text{-}\mathsf{PRF}}$ following from definition 2.2, where the pseudorandom function property of the Extract and Expand functions is shown in [4, 5]. The collision resistance of Extract immediately follows from the collision resistance of SHA [18]. Finally, the term $\mathbf{Adv}^{\mathsf{AES},\mathcal{A}_2}_{\mathsf{CCA}}$, as per [6], refers to the security of the underlying AEAD scheme used in the hybrid PKE. We can reduce the tightness of origin privacy if we relax the definition by providing a signing oracle instead of providing the adversary with all secret keys. Using the same key generation technique we used in the one-more unforgeability proof and the perfect blindness of the key-blinding Schnorr signature, we can reduce the quadratic loss $n_C \cdot n_O \cdot \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{DDH}}$ to $n_O \cdot \mathbf{Adv}^{\mathcal{A}_4}_{\mathsf{DDH}}$. This would suggest that RlP should be instantiated with a key-blinding signature scheme that achieves perfect or statistical indistinguishability.

7 CONCLUSION

We have formalized the syntax and security notions for the rate-limited Privacy Pass in a comprehensive definition. We also provided a generic construction on top of the component-dependent scheme in [20] proposed for IETF Standardization. The following security analysis requires no random oracle model. We also stress that the construction achieves strong privacy notions. We showed client privacy with malicious key generation and full exposure of client keys. On the other hand, we allow the adversary in the origin privacy experiment to choose the target client and maliciously generate the origin key. We do not consider privacy properties for aborting executions and leave the investigation of these cases as an exciting scope for future research.

We achieved the security results using two privacy notions for the key-blinding signature scheme. The notions formalized in our work never appeared in any previous works. The two definitions not only help formulate the generic framework for RlP out of the intuition from [20] but also can be of independent interest.

This work leads to some future research directions. One of the interesting questions is how to construct the key-blinding building block to complete the post-quantum RlP instantiation.

REFERENCES

[1] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. 2022. Practical, Round-Optimal Lattice-Based Blind Signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 39–53. https://doi.org/10.1145/3548606.3560650

[2] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. 2021. Analysing the HPKE Standard. In *Advances in Cryptology – EUROCRYPT 2021*, Anne Canteaut and François-Xavier Standaert (Eds.). Springer International Publishing, Cham, 87–116.

[3] Mila Anastasova, Panos Kampanakis, and Jake Massimo. 2022. PQ-HPKE: Post-Quantum Hybrid Public Key Encryption. Cryptology ePrint Archive, Paper 2022/414. https://eprint.iacr.org/2022/414 https://eprint.iacr.org/2022/414.

[4] Mihir Bellare. 2006. New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In *Advances in Cryptology - CRYPTO 2006*, Cynthia Dwork (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 602–619.

[5] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. 1996. Keying Hash Functions for Message Authentication. In *Advances in Cryptology — CRYPTO '96*, Neal Koblitz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–15.

[6] Mihir Bellare and Björn Tackmann. 2016. The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology – CRYPTO 2016*, Matthew Robshaw and Jonathan Katz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 247–276.

[7] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. 2018. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. , 353-367 pages. https://doi.org/10.1109/EuroSP.2018.00032
[8] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. 2020. ZEXE: Enabling Decentralized Private Computation. , 947-964 pages. https://doi.org/10.1109/SP40000.2020.00050
[9] Cloudflare. 2022. What is rate limiting? | Rate limiting and bots. https://www.cloudflare.com/learning/bots/what-is-rate-limiting/.
[10] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *Proceedings on Privacy Enhancing Technologies* 2018 (06 2018), 164–180. https://doi.org/10.1515/popets-2018-0026
[11] F. Denis, E. Eaton, T. Lepoint, and C.A. Wood. 2021. Key Blinding for Signature Schemes. https://www.ietf.org/archive/id/draft-irtf-cfrg-signature-key-blinding-02.html.
[12] Frank Denis, Frederic Jacobs, and Christopher A. Wood. 2022. *RSA Blind Signatures*. Internet-Draft draft-irtf-cfrg-rsa-blind-signatures-07. IETF Secretariat. https://www.ietf.org/archive/id/draft-irtf-cfrg-rsa-blind-signatures-07.txt https://www.ietf.org/archive/id/draft-irtf-cfrg-rsa-blind-signatures-07.txt.
[13] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. 2023. Security Analysis of Signature Schemes with Key Blinding. Cryptology ePrint Archive, Paper 2023/380. https://eprint.iacr.org/2023/380 https://eprint.iacr.org/2023/380.
[14] Edward Eaton, Sajin Sasy, and Ian Goldberg. 2022. Improving the Privacy of Tor Onion Services. In *Applied Cryptography and Network Security*, Giuseppe Ateniese and Daniele Venturi (Eds.). Springer International Publishing, Cham, 273–292.
[15] Edward Eaton, Douglas Stebila, and Roy Stracovsky. 2021. Post-quantum Key-Blinding for Authentication in Anonymity Networks. In *Progress in Cryptology – LATINCRYPT 2021*, Patrick Longa and Carla Ràfols (Eds.). Springer International Publishing, Cham, 67–87.
[16] J. Finn. 2022. HBO Max While Traveling: What To Know Before Leaving Home. https://streamingbetter.com/hbo-max-while-traveling https://streamingbetter.com/hbo-max-while-traveling.
[17] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. 2016. Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys. In *Public-Key Cryptography – PKC 2016*, Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 301–330.
[18] Henri Gilbert and Helena Handschuh. 2004. Security Analysis of SHA-256 and Sisters. In *Selected Areas in Cryptography*, Mitsuru Matsui and Robert J. Zuccherato (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 175–193.
[19] Google. 2019. Rate-limiting strategies and techniques. https://cloud.google.com/architecture/rate-limiting-strategies-techniques.
[20] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. 2022. *Rate-Limited Token Issuance Protocol*. Internet-Draft draft-ietf-privacypass-rate-limit-tokens-00. IETF Secretariat. https://www.ietf.org/archive/id/draft-ietf-privacypass-rate-limit-tokens-00.txt https://www.ietf.org/archive/id/draft-ietf-privacypass-rate-limit-tokens-00.txt.
[21] Dennis Hofheinz and Eike Kiltz. 2007. Secure Hybrid Encryption from Weakened Key Encapsulation. In *Advances in Cryptology - CRYPTO 2007*, Alfred Menezes (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 553–571.
[22] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. 2016. Breaking Symmetric Cryptosystems Using Quantum Period Finding. In *Advances in Cryptology – CRYPTO 2016*, Matthew Robshaw and Jonathan Katz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 207–237.
[23] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. 1997. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. RFC Editor. http://www.rfc-editor.org/rfc/rfc2104.txt http://www.rfc-editor.org/rfc/rfc2104.txt.
[24] H. Krawczyk and P. Eronen. 2010. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. RFC 5869. RFC Editor. http://www.rfc-editor.org/rfc/rfc5869.txt http://www.rfc-editor.org/rfc/rfc5869.txt.
[25] Benjamin Lipp. 2020. An Analysis of Hybrid Public Key Encryption. Cryptology ePrint Archive, Paper 2020/243. https://eprint.iacr.org/2020/243 https://eprint.iacr.org/2020/243.
[26] Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. 2016. On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks. In *Information Security and Cryptology - ICISC 2015*, Soonhak Kwon and Aaram Yun (Eds.). Springer International Publishing, Cham, 20–35.
[27] Netflix. 2022. Traveling or moving with Netflix. https://help.netflix.com/en/node/24853 https://help.netflix.com/en/node/24853.
[28] NIST. 2022. NIST Announces First Four Quantum-Resistant Cryptographic Algorithms. https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms
[29] Tatsuaki Okamoto. 2007. Authenticated Key Exchange and Key Encapsulation in the Standard Model. In *Advances in Cryptology – ASIACRYPT 2007*, Kaoru Kurosawa (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 474–484.
[30] Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A Wood. 2023. Post-Quantum Privacy Pass via Post-Quantum Anonymous Credentials. Cryptology ePrint Archive, Paper 2023/414. https://eprint.iacr.org/2023/414 https://eprint.iacr.org/2023/414.
[31] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. 2018. Tightly-Secure Key-Encapsulation Mechanism in the Quantum Random Oracle Model. In *Advances in Cryptology – EUROCRYPT 2018*, Jesper Buus Nielsen and Vincent Rijmen (Eds.). Springer International Publishing, Cham, 520–551.
[32] Victor Shoup. 2001. A Proposal for an ISO Standard for Public Key Encryption. Cryptology ePrint Archive, Paper 2001/112. https://eprint.iacr.org/2001/112 https://eprint.iacr.org/2001/112.
[33] Riad S. Wahby, Dan Boneh, Christopher Jeffrey, and Joseph Poon. 2020. An Airdrop that Preserves Recipient Privacy. In *Financial Cryptography and Data Security*, Joseph Bonneau and Nadia Heninger (Eds.). Springer International Publishing, Cham, 444–463.
[34] C. A. Wood. 2022. Rate-Limited Privacy Pass. https://www.csail.mit.edu/event/rate-limited-privacy-pass https://www.csail.mit.edu/event/rate-limited-privacy-pass.