Completing Policy-based Anonymous Tokens: Private Bits, Public Metadata and more...

David Kretzler
Huawei Heisenberg Research Center
Germany
david.kretzler@huawei.com

Yong Li
Huawei Heisenberg Research Center
Germany
yong.li1@huawei.com

Codrin Ogreanu
Huawei Heisenberg Research Center
Germany
codrin.ogreanu@gmail.com

Abstract

Anonymous tokens are cryptographic protocols for restricting the access to online resources to eligible users. After proving eligibility to the token issuer, the client receives a set of tokens. Later, it can prove eligibility to a resource provider by sending one of the tokens received from the issuer. The anonymous token protocol ensures that the resource provider cannot link received tokens to their issuance, even if it colludes with the token issuer. Recently, Faut et al. (EuroS&P'25) introduced the concept of policy-based anonymous tokens, in which an issuer provides a single pre-token to a client, who can locally derive multiple tokens according to a publicly announced policy. The major advantage of policy-based tokens is that the communication complexity of the issuance phase is constant. While the work of Faut et al. constitutes a promising step in a new direction, their protocol still lacks several desirable properties known from standard anonymous tokens – most notably, the ability to bind a pre-token and all tokens derived from it to a private metadata bit or a publicly known metadata string.

In this work, we present a new framework for policy-based anonymous token schemes in the random oracle model. Our framework includes two concretely practical constructions – one based on equivalence class signatures and one on algebraic MACs – as well as a communication-optimized, though less practical, construction based on zkSNARKs. All three constructions can be configured to support private metadata bits, public metadata, or both. We formalize the notion of policy-based anonymous tokens with a private metadata bit and public metadata, and we prove security of the two primary constructions: the equivalence-class-signature-based scheme and the algebraic-MAC-based scheme. Finally, we provide an experimental evaluation and comparison of all our constructions alongside the most relevant related work. Our results demonstrate that our two primary constructions achieve significant efficiency improvements over the scheme of Faut et al., both in terms of computation communication.

CCS Concepts

• **Security and privacy → Cryptography**; **Privacy-preserving protocols**.

Keywords

Anonymous Tokens, Policy-based Tokens, Private Metadata Bit, Public Metadata

1 Introduction

Anonymous token schemes as introduced by [17] limit the access to resources to eligible users without identifying the user upon resource access. These schemes typically involve three parties: a client, an issuer, and a verifier. Initially, the client engages in an interactive issuance protocol with the issuer to obtain a token. During this protocol, the client proves that it is eligible to access the protected resource, e.g., by proving human-hood via a CAPTCHA, and the issuer issues a set of tokens. Later, the client presents the token to the verifier in what is called the token redemption. The verifier confirms the token's authenticity and, if valid, grants access to a particular resource or service. To prevent multiple uses of the same token, the verifier usually checks whether a token has been redeemed before. The two fundamental security requirements for anonymous tokens are accountability and unlinkability: accountability guarantees that only tokens created by the legitimate issuer are accepted as valid and unlinkability ensures that the issuer and verifier cannot link the issuance and redemption sessions of a token.

A broad range of anonymous token schemes have been proposed [1, 2, 4, 5, 10, 11, 17, 20, 21, 26, 27, 30, 37] offering different feature sets and trade-offs. Some constructions [11, 17, 27] assume the verifier and issuer to be the same party by requiring possession of the token issuance key for the token verification. Other schemes [1, 2, 4, 5, 10, 20, 21, 26, 30, 37] support public verifiability of tokens, allowing the separation between the issuer and the verifier. Another line of work allows the inclusion of metadata into tokens that can be public [1, 20, 37] or private [2, 4, 11, 27]. Public metadata encodes additional information such as revocation or usage policies, whereas private metadata allows issuers to embed hidden flags – e.g., to silently invalidate a token request without revealing rejection to the client. By hiding whether a token request was successful, it becomes harder for adversaries to infer strategies for bypassing the eligibility verification process. We note, however, that the inclusion of metadata weakens the anonymity guarantees, as a token's anonymity set is limited to the other tokens that share the same metadata.

Recently Faut et al. [21] introduced the concept of *policy-based anonymous tokens (PBAT)*. In PBAT schemes, the client receives a single pre-token during the issuance phase from which it locally derives several complete tokens. Each complete token is derived from a specific tag. The verifier accepts a token only if the corresponding tag is contained in a predefined policy, modeled as a list of allowed tags. For example, a policy may allow each client to generate up to ten tokens per day by including tags formed by concatenating the current date with a counter taking values from zero to nine. PBAT schemes ensure that each pre-token produces one unique token per tag such that clients can indeed only spend one token per tag that is part of the policy. The major advantage of PBAT is that the complexity of the token issuance is constant; in particular, it is independent of the number of tokens that are derived from a single pre-token. This property is particularly useful in settings, in which there is a single token issuer but a multitude of token verifiers, e.g., offering different services or resources. While the work of [21] constitutes a first step into a promising new direction, it lacks some desirable properties. Most importantly, it supports neither public nor private metadata. The utilization of public metadata is particularly interesting for policy-based tokens, as there is no inherent revocation date for pre-tokens, which can be added with public metadata.

1.1 Contribution.

In this work, we extend the research on policy-based anonymous tokens by proposing a framework of protocols offering public verifiability, private metadata bits, public metadata or any combination of these properties. More precisely, we present two primary constructions, one that is publicly verifiable and one that is not, and extensions for both protocols allowing for private metadata bits and public metadata. Furthermore, we discuss further extensions augmenting the constructions with non-interactiveness and non-transferability.

Following the approach of [21], we base our publicly verifiable construction on equivalence class signatures (EQS) [22]. However, we replace the token derivation and verification mechanism with a simpler and more efficient alternative. Our second construction employs the same token derivation and verification mechanism but combines it with the algebraic message authentication code (MAC) of [18]. While this construction sacrifices the public verifiability property, it does not rely on bilinear pairings, the basic building block of known secure instantiations of equivalence class signatures. Therefore, it provides much better efficiency, both in terms of communication and computation. In addition to the two primary protocols, we include a third construction based on zkSNARKs (zero-knowledge succinct non-interactive argument of knowledge) into our framework. This protocol provides public verifiability, can also be extended with private metadata bits and public metadata, and offers significantly better communication complexity than our EQS-based construction. However, due to the high computational overhead of creating zkSNARKs, the zkSNARK-based construction is not practical and, hence, constitutes just a theoretical result.

We formally define the notion of *policy-based anonymous tokens with private metadata bit and public metadata* and prove security of the two primary protocols (with the extensions for private metadata bits and public metadata). As the security of the only known secure instantiation of equivalence class signatures [22][1] and the MAC scheme of [18] only holds in the generic group model, it follows that security of our two primary protocols only holds in the generic group model as well. Furthermore, we require the random oracle model to prove security.

[1]It has been shown by [3] that equivalence class signature schemes cannot be proven secure under standard assumptions.

We have implemented and benchmarked all our constructions – with and without private metadata bits and public metadata – as well as the most important related work for comparison. The latter includes the state-of-the-art standard anonymous token scheme, PrivacyPass, and the construction of [21]. Our benchmarks show that our MAC-based construction is competitive with Privacy Pass (which also does not offer public verifiability) and our EQS-based construction outperforms the construction proposed by [21]. For token redemption, our EQS-based construction reduces communication to below 45 % and runtime to below 50 % of those in [21].

1.2 Technical Overview

All our constructions follow the same blueprint. In the issuance phase, the client samples a secret key $\mathsf{sk}_c \xleftarrow{\$} \mathbb{Z}_p$ and receives some authentication information $\mathsf{Auth}[\mathsf{sk}_c]$ for its secret key from the issuer. The issuer does not learn sk_c when issuing $\mathsf{Auth}[\mathsf{sk}_c]$. To redeem a token for tag τ, the client hashes the tag $T = \mathcal{H}_2(\tau)$ with a hash function $\mathcal{H}_2$ modeled as a random oracle, derives the token $\delta = \mathsf{sk}_c T$ and computes a witness ω proving possession of authentication information $\mathsf{Auth}[\mathsf{sk}_c]$ and secret sk_c such that $\mathsf{dlog}_T(\delta) = \mathsf{sk}_c$. The witness does not disclose any information about $\mathsf{Auth}[\mathsf{sk}_c]$ or sk_c. Note that the token computation constitutes a pseudo random function (PRF) evaluation on input τ under key sk_c. Our constructions differ in the realization of the authentication information and the witness computation.

Construction based on digital signatures and zkSNARKs. As a warm-up, we consider a simple construction based on digital signatures and zkSNARKs. During issuance, the client sends a client public key $\mathsf{pk}_c = \mathsf{sk}_c G$ to the issuer and receives a signature σ on pk_c, which represents the authentication information $\mathsf{Auth}[\mathsf{sk}_c]$. To authenticate the token $\delta = \mathsf{sk}_c T$ the client computes a zkSNARK that proves knowledge of a signature σ and a public key pk_c such that σ is a valid signature on pk_c and $\mathsf{dlog}_G(\mathsf{pk}_c) = \mathsf{dlog}_T(\delta)$.

As the issuer is trusted to issue tokens only to eligible clients, it can also be trusted to perform the zkSNARK's setups without exploiting the toxic waste that falls off during the setup. This is because the toxic waste only permits forging proofs – and thus additional tokens – which can also be done with the issuer's secret key. While the resulting scheme offers public verifiability and is highly efficient in terms of communication, it comes with a substantial overhead in terms of computation due to the utilization of general-purpose zkSNARKs.

Construction based on equivalence class signatures. An alternative to relying on zkSNARKs is to make use of equivalence class signatures for the pre-token authentication mechanism. Equivalence class signatures enable a signer to issue a constant-size signature σ on a message $(M_1, M_2, \ldots, M_n)$ that can be re-randomized by the signature holder into a signature σ^* on a message $(M_1^*, M_2^*, \ldots, M_n^*)$ such that $M_i^* = rM_i$ for $i \in [n]$ and $r \xleftarrow{\$} \mathbb{Z}_p$.

In the EQS-based construction, the client requests a pre-token by sending a public key $\mathsf{pk}_c = \mathsf{sk}_c G$ to the issuer. The issuer generates a message $\mathsf{msg} = (vG_1, v \cdot \mathsf{pk}_c)$ with $v \xleftarrow{\$} \mathbb{Z}_p^*$, creates an EQS σ on msg and sends (msg, σ) to the client. The tuple (msg, σ) represents the authentication information $\mathsf{Auth}[\mathsf{sk}_c]$. To redeem a token $\delta = \mathsf{sk}_c T$, the client selects $r \xleftarrow{\$} \mathbb{Z}_p^*$, computes a re-randomized signature σ^* on message $\mathsf{msg}^* = (rM_1, rM_2) = (M_1^*, M_2^*)$ and sends $(\mathsf{msg}^*, \sigma^*)$ together with a zero-knowledge proof that $\mathsf{dlog}_T(\delta) = \mathsf{dlog}_{M_1^*}(M_2^*)$. The re-randomization of the message and signature ensures that the token cannot be linked to the issuance of its pre-token.

The EQS-based construction combines public verifiability with high computational efficiency, as zero-knowledge proofs of equality of discrete logarithms can be instantiated efficiently using generalized Schnorr proofs [34]. Its main drawbacks are the reliance on a bilinear pairing—which is less efficient than standard elliptic curves—and a higher communication complexity compared to the zkSNARK-based construction.

Construction based on algebraic MACs. We observe that it is possible to authenticate the client key sk_c with message authentication codes (MACs) instead of signatures if public verifiability of tokens is not required. Therefore, we build a construction trading public verifiability for increased efficiency that is based on the algebraic MAC construction of [18]. To request a token, the client sends a public key $\mathsf{pk}_c = \mathsf{sk}_c G$ to the issuer, who responds with a MAC $(M_1, M_2) = (vG, (\mathsf{sk}_1^s \cdot v)G + (\mathsf{sk}_2^s \cdot v)\mathsf{pk}_c)$, where $(\mathsf{sk}_1^s, \mathsf{sk}_2^s) \in \mathbb{Z}_p^2$ is the issuer's secret MAC key and $v \xleftarrow{\$} \mathbb{Z}_p^*$ is freshly sampled. Since the client cannot verify the MAC in the same way it would verify a signature, the issuer includes a zero-knowledge proof proving that the MAC was generated correctly, i.e., using the issuer's secret MAC key. For this proof, it is necessary to establish a reference to the intended MAC keys. To this end, the issuer publishes the public key $\mathsf{pk}_2^s = \mathsf{sk}_2^s G$ and a Pedersen commitment C to sk_1^s – the value $\mathsf{sk}_1^s G$ cannot be published for security reasons. For the redemption, the client samples $r \xleftarrow{\$} \mathbb{Z}_p^*$, re-randomizes the MAC to $(M_1^*, M_2^*) = (rM_1, rM_2)$ and the public key to $\mathsf{pk}_c^* = \mathsf{sk}_c M_1^*$, computes the token $\delta = \mathsf{sk}_c T$ and a zero-knowledge proof for $\mathsf{dlog}_{M_1^*}(\mathsf{pk}_c^*) = \mathsf{dlog}_T(\delta)$. The client sends the re-randomized MAC, the token and the zero-knowledge proof to the verifier. The verifier can derive the re-randomized public key itself by computing $\mathsf{pk}_c^* = \frac{1}{\mathsf{sk}_2^s}(M_2^* - \mathsf{sk}_1^s M_1^*)$. Equivalently to the EQS-based construction, the re-randomization of the MAC ensures that the token cannot be linked to the issuance of its pre-token. We note that all the zero-knowledge proofs utilized by this construction can be instantiated efficiently with generalized Schnorr proofs.

Including public metadata. All constructions can be modified to incorporate public metadata without any increase in the communication complexity. In the zkSNARK-based construction, public metadata is incorporated by appending the metadata to the client's public key before signing it.

In the EQS-based construction, we incorporate public metadata by extending the message $\mathsf{msg} = (M_1, M_2)$ with a third component $M_3 = \mathcal{H}_1(\mathsf{md}) \cdot M_1$, using a hash function $\mathcal{H}_1$ modeled as a random oracle. This binds all tokens derived from a particular pre-token to the pre-token's metadata, as the EQS-signature authenticates only messages of the form $(M_1^*, M_2^*, M_3^*) = (rM_1, rM_2, rM_3)$ for $r \in \mathbb{Z}_p$. For all these messages it holds that $M_3^* = \mathcal{H}_1(\mathsf{md})M_1^*$. We note that is it not necessary to send M_3 during token issuance or M_3^* during token redemption. Since the metadata is public, it is possible to compute M_3 from M_1 and M_3^* from M_1^*. In the public metadata extension of the MAC-based construction, we make use of a third MAC key component sk_3^s and publish $\mathsf{pk}_3^s = \mathsf{sk}_3^s G$. MACs are then issued as $(M_1, M_2) = (vG, (\mathsf{sk}_1^s + \mathsf{sk}_3^s \cdot \mathcal{H}_1(\mathsf{md}))R + (\mathsf{sk}_2^s \cdot v)\mathsf{pk}_c)$ and verified accordingly. By designing the issuance proof to prove correct issuance of MACs based on pk_3^s without having to prove knowledge of sk_3^s, we avoid an increase in the communication complexity during the issuance. Since, the verifier can compute the metadata-dependent term itself and subtract it from M_2^* when computing the re-randomized public key pk_c^*, we also avoid additional communication during the redemption phase.

Private metadata bit. To include a private metadata bit in the zkSNARK-based construction, we extend the issuer public key, which was just a digital signature key before, with two private bit public keys $\mathsf{pk}_{\mathsf{pb}}^0 = \mathsf{sk}_{\mathsf{pb}}^0 G$ and $\mathsf{pk}_{\mathsf{pb}}^1 = \mathsf{sk}_{\mathsf{pb}}^1 G$. When issuing a token with private bit ok, the issuer samples $v \xleftarrow{\$} \mathbb{Z}_p^*$, computes $R = vG_1$ and $X = v\mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}}$ and appends (R, X) to the message that is to be signed, i.e., the client's public key and the public metadata. The issuer then signs the message and sends (R, X) together with signature σ to the client. Since the client must be assured that (R, X) has been computed correctly, the issuer also includes zero-knowledge proof of knowledge attesting that it knows v such that $R = vG$ and X is either equal to $(v\mathsf{pk}_{\mathsf{pb}}^0)$ or $(v\mathsf{pk}_{\mathsf{pb}}^1)$. During the token redemption, the client re-randomizes (R, X) to $(R^* = rR, X^* = rX)$ with $r \xleftarrow{\$} \mathbb{Z}_p^*$ and sends (R^*, X^*) together with its zkSNARK to the verifier, which verifies whether $X^* = \mathsf{sk}_{\mathsf{pb}}^0 R^*$ or $X^* = \mathsf{sk}_{\mathsf{pb}}^1 R^*$. In the zkSNARK, the client additionally proves that it knows r such that the signature is valid with respect to $(\frac{1}{r}R^*, \frac{1}{r}X^*)$.

In the EQS-based construction, we also make use of two private bit public keys $\mathsf{pk}_{\mathsf{pb}}^0 = \mathsf{sk}_{\mathsf{pb}}^0 G$ and $\mathsf{pk}_{\mathsf{pb}}^1 = \mathsf{sk}_{\mathsf{pb}}^1 G$ and include a private bit tuple (R, X) into the signed message. However, we can use the first message component $M_1 = vG$ as R and only add a single element $(X = v \cdot \mathsf{pk}_{\mathsf{pb}}^b)$ as fourth element to the signed message. As before, the issuer has to include a zero-knowledge proof ensuring that X is computed based on either $\mathsf{pk}_{\mathsf{pb}}^0$ or $\mathsf{pk}_{\mathsf{pb}}^1$. For redemption, the client additionally transmits the re-randomized X^* and the verifier can check whether $X^* = \mathsf{sk}_{\mathsf{pb}}^0 M_1^*$ or $X^* = \mathsf{sk}_{\mathsf{pb}}^1 M_1^*$.

In the MAC-based construction, the issuer samples two distinct MAC keys, $\mathsf{sk}_0^s = (\mathsf{sk}_{0,1}^s, \mathsf{sk}_{0,2}^s, \mathsf{sk}_{0,3}^s)$ and $\mathsf{sk}_1^s = (\mathsf{sk}_{1,1}^s, \mathsf{sk}_{1,2}^s, \mathsf{sk}_{1,3}^s)$, and uses $\mathsf{sk}_{\mathsf{ok}}^s$ to encode bit ok into the pre-token. The issuer publishes the public keys and commitments for both of the MAC keys and proves for each MAC that it has been issued with either of the two MAC keys. During redemption, the verifier attempts verification using both sk_0^s and sk_1^s. If verification succeeds with $\mathsf{sk}_{\mathsf{ok}}^s$, the verifier outputs private bit ok.

1.3 Related Work

Anonymous tokens have been the subject of extensive research [1, 2, 4, 5, 11, 17, 20, 21, 26, 27, 30, 37]. One line of work [17, 27, 30, 37] bases their constructions on Oblivious Pseudo Random Functions (OPRFs). An OPRF is two-party protocol allowing a client to obtain a PRF evaluation on its input computed with the server's secret key without disclosing the input to the server. Davidson et al. [17] introduce the concept of anonymous tokens, without public verifiability or any kind of metadata. In [27] and [37], the authors extend the work of [17] by incorporating the private metadata bit property in both, and the public metadata property and public verifiability in the latter. While the construction of [37] requires bilinear pairings to provide public verifiability, the work of [30] proposes the first OPRF-based construction offering public verifiability without relying on bilinear pairings.

An alternative approach to OPRFs adopted by [1, 2, 4, 26] is to base anonymous tokens on blind signatures as introduced by [13, 14]. In blind signature schemes, the signer issues a signature without learning the signed message. Anonymous tokens based on blind signatures are inherently publicly verifiable. In [4], Benhamouda et al. propose a construction with private metadata bits based on blind Schnorr signatures [23, 34]. Karantaidou et al. [26] utilizes blind BLS multi-signatures [6, 8, 9] and Snowblind [16] to build anonymous tokens with a decetralized token issuance. Amjad et al. [1] present a construction with public metadata that is based on RSA signatures [33]. Baldimitsi et al. [2] proposes so-called non-interactive anonymous tokens based on non-interactive blind signatures [25]. Non-interactive anonymous tokens allow the issuer of a token to issue tokens to known clients without requiring the clients to send a token request.

Another line of work [11, 20] builds anonymous tokens without public verifiability based on algebraic MACs. In both works, the MACs are realized as BBS signatures [7] in a pairing-free group. Chase et al. [11] identify a weakness in the definition of the private bit property of [27, 37] and present a MAC-based construction addressing the identified weakness. Durak et al. [20] propose a non-transferable anonymous token construction that deters clients from transferring tokens between each other by binding each token to a valuable insurance secret that needs to be known for the token redemption – a technique known from anonymous credentials [28, 31]. Their work can be extended to support public metadata and public verifiability. For the latter, they use bilinear pairings.

Anonymous counting tokens [5] are a special type of tokens ensuring that each user can only get one token per context and, hence, enabling verifiers to count the number of users that redeemed a token for a particular context. Benhamouda et al. [5] propose several constructions for anonymous counting tokens based on OPRFs or equivalence class signatures. We note that policy-based tokens can be used to instantiate counting tokens by defining a single tag for each context. However, the constructions of [5] cannot be used to instantiate policy-based tokens as they require communication per issued token, while policy-based tokens require the issuance communication to be constant.

Policy-based tokens have been introduced by Faut et al. [21]. The authors formally define the notion of policy-based anonymous tokens, introduce a new cryptographic primitive called group verifiable random function (GVRF) and show how policy-based tokens can be instantiated generically from GVRFs. Group verifiable random functions allow a group manager to issue distinct member keys that can be used to evaluate a verifiable random function (VRF) on behalf of the group. The GVRF ensures that different members produce distinct yet valid outputs and that these outputs cannot be linked to the issuance of the corresponding member key. Policy-based tokens can be constructed straightforwardly from GVRFs by issuing pre-tokens as member keys and deriving tokens as GVRF evaluations on the tags permitted by the policy. The authors present an instantiation of GVRFs based on equivalence class signatures [22] and the Dodis-Yampolskiy VRF [19]. In their construction, member keys are issued in form of equivalence class signatures on a generator-public key pair (G, xG). To evaluate the function, the generator-public key pair is re-randomized to (vG, xvG) and the token is computed as a Dodis-Yampolskiy VRF on the tag under secret key x. The authors modify the VRF such that the evaluation is verifiable with respect to the randomized public key (vG, xvG) instead of the original one. The resulting policy-based anonymous token scheme avoids the use of zero-knowledge proofs in the pre-token issuance and token redemption, and can be proven secure without relying on the random oracle model. Our publicly verifiable construction utilizes EQS in the same way, i.e., to authenticate a re-randomized generator-public key pair, but employs a simpler PRF for token computation and achieves verifiability via a Schnorr-style zero-knowledge proof. Although the PRF and the zero-knowledge proof used in our construction can only be proven secure in the random oracle model, our construction significantly outperforms that of [21] (cf. Section 6). Additionally, our framework includes extensions for private metadata bits and public metadata while [21] does not offer such features. Although our construction implicitly contains a GVRF, we chose to instantiate PBAT directly to keep the presentation concise and simple.

2 Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$, the set $\{1, ..., b\}$ by $[b]$, and the set $\{a, a+1, \ldots, b-1, b\}$ by $[a, b]$. When considering sets, we always assume them to be ordered multisets. We denote the i-th element of a set $\mathcal{S}$ by $\mathcal{S}[i]$, a negligible function by negl, computational indistinguishability of two distributions by $\approx_c$, probabilistic polynomial time algorithms by PPT, the execution of a PPT algorithm $\mathcal{A}$ on input b by $a \leftarrow \mathcal{A}(b)$ and, finally, the execution of a PPT algorithm $\mathcal{A}$ with fixed random tape r by $a \leftarrow \mathcal{A}(b; r)$.

We consider both standard (pairing-free) and pairing-friendly prime-order cyclic groups. Algorithm $(p, \mathbb{G}) \leftarrow \mathsf{GGen}(1^\lambda)$ generates a standard group $\mathbb{G}$ of order p, where p is a λ-bit prime. We assume the decisional Diffie-Hellman (DDH) assumption to hold for GGen. Algorithm $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathrm{e}) \leftarrow \mathsf{BGGen}(1^\lambda)$ generates a bilinear pairing group, in which p is a λ-bit prime, $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ have order p, and e is an efficiently computable map such that $\mathrm{e}(aU, bV) = (ab)\mathrm{e}(U, V)$ for all $(U, V, a, b) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_p \times \mathbb{Z}_p$ and $\mathrm{e}(G_1, G_2) \neq 1$ for all generators $G_1 \in \mathbb{G}_1$ and $G_2 \in \mathbb{G}_2$. We denote by $\mathbb{G}^* = \mathbb{G} \setminus \{1_\mathbb{G}\}$ the set of all generators of $\mathbb{G}$ for $\mathbb{G} \in \{\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2\}$. In this work, we consider Type-2 or Type-3 pairings, i.e., pairings, in which no efficient homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is known. We assume the external Diffie-Hellman (XDH) assumption to hold for BGGen which implies that the discrete logarithm is hard in groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ and the decisional Diffie-Hellman (DDH) assumption holds for BGGen with respect to Diffie-Hellman tuples sampled from $\mathbb{G}_1$. We denote the maximal advantage of an adversary in distinguishing DDH tuples in $\mathbb{G}_1$ by $\mathsf{Adv}_{\mathsf{ddh}}$.

2.1 Zero Knowledge Proofs of Knowledge

A zero-knowledge proof of knowledge enables a prover to convince a verifier that it knows a witness w for a statement s of a NP-language $\mathcal{L}_\mathcal{R}$ with associated relation $\mathcal{R}$ such that $(s, w) \in \mathcal{R}$. We denote the proof relation by $\mathsf{ZKP}_{\mathcal{R}} = \mathsf{ZKP}\{w : \mathcal{R}(s, w) = 1\}$ and define non-interactive zero-knowledge proofs in the random oracle model as follows:

Definition 1. *Let $\mathcal{R}$ be a NP-relation, $\mathcal{L}_{\mathcal{R}} = \{s : (s, w) \in \mathcal{R}\}$ be the associated NP-language and $\mathcal{H}$ be a global random oracle. A non-interactive zero-knowledge proof of knowledge for relation $\mathcal{R}$ consists of the following two polynomial time algorithms* (Prove, Verify) *with access to a random oracle $\mathcal{H}$.*

- $\mathsf{Prove}(w, s) \rightarrow \pi$*: a probabilistic algorithm that takes as input a witness w and a statement s, and outputs a proof π.*
- $\mathsf{Verify}(\pi, s) \rightarrow 0/1$*: a deterministic algorithm that takes as input a proof π and a statement s, and outputs either 1 accepting the proof or 0 rejecting it.*

The algorithms satisfy the following properties:

- *Completeness: For all $\lambda \in \mathbb{N}$ and $(s, w) \in \mathcal{R}$, it holds that*

$$\Pr\left[\mathsf{Verify}(\pi, s) = 1 : \quad \pi \leftarrow \mathsf{Prove}(w, s) \quad\right] = 1.$$

- *Knowledge soundness: For every $\lambda \in \mathbb{N}$, PPT adversary $\mathcal{A}$ and random tape r, there exists a PPT extractor $\mathcal{E}$ controlling oracle $\mathcal{H}$ with black-box access to $\mathcal{A}$ and a negligible function* negl *such that*

$$\Pr\left[\begin{array}{lcl} \mathsf{Verify}(\pi, s) = 1 & : & (s, \pi) \leftarrow \mathcal{A}(1^\lambda; r) \\ \wedge (s, w) \notin \mathcal{R} & & w \leftarrow \mathcal{E}^{\mathcal{H},\mathcal{A}}(1^\lambda, r) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

  We assume extractor $\mathcal{E}$ to return $\perp$ if it is not successful, i.e., we assume that $w = \perp$ if $(s, w) \notin \mathcal{R}$.
- *Zero-knowledge: There exists a PPT simulator $\mathcal{S}$ controlling oracle $\mathcal{H}$ such that for every $(s, w) \in \mathcal{R}$ it holds that*

$$\{\mathsf{Prove}(w, s)\} \approx_c \{\mathcal{S}^{\mathcal{H}}(1^\lambda, s)\}.$$

For our MAC-based construction, we require an additional property that we call statement-oblivious knowledge soundness. In particular, we require that the extractor, which sees only the proof, can extract the witness and the statement under which the proof verifies (if any). Formally, we define this property as follows:

Definition 2. *A zero-knowledge proof system as defined in Definition 1 satisfies statement-oblivious knowledge soundness if for every $\lambda \in \mathbb{N}$, pair of PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$ and random tape r, there exists a PPT extractor $\mathcal{E}_{\mathrm{Ob}}$ controlling oracle $\mathcal{H}$ with black-box access to $\mathcal{A}_1$ and a negligible function* negl *such that*

$$\Pr\left[\begin{array}{lcl} \mathsf{Verify}(\pi, s) = 1 \wedge & & (\mathsf{state}, \pi) \leftarrow \mathcal{A}_1(1^\lambda; r) \\ ((s, w) \notin \mathcal{R} & : & (s', w) \leftarrow \mathcal{E}_{\mathrm{Ob}}^{\mathcal{H},\mathcal{A}}(1^\lambda, r) \\ \vee s \neq s') & & (s) \leftarrow \mathcal{A}_2(\mathsf{state}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We assume extractor $\mathcal{E}_{\mathrm{Ob}}$ to return $(\perp, \perp)$ if it is not successful, i.e., we assume that $(s', w) = (\perp, \perp)$ if $\mathsf{Verify}(\pi, s') = 0$ or $(s', w) \notin \mathcal{R}$.

The intuition for this property in Fiat Shamir-transformed sigma protocols is that the computation of the challenge requires a random oracle query containing the statement, which is observed by the extractor. As the random oracle samples the challenge randomly, the probability of guessing a correct challenge for proof π and statement s without querying the oracle is negligible in the oracle's output domain.

2.2 Equivalence Class Signatures

We employ equivalence class signatures as introduced by [22]. While we adopt most of their definition verbatim, we explicitly specify a security game for the unforgeability property. Throughout, we assume that all algorithms—except for public parameter generation—implicitly take the public parameters as their first input.

Definition 3. *An equivalence class signature scheme (EQS) for messages of size $\ell > 1$ with relation*

$$\mathcal{R} = \{(U, V) \in ((\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell) \mid \exists a \in \mathbb{Z}_p^* : V = aU\}$$

is defined via the following PPT algorithms:

- $\mathsf{BGGen}(1^\lambda) \rightarrow \mathsf{pp}$*: a bilinear-group generation algorithm that takes as input the security parameter λ and outputs the public parameters, i.e., the description of a bilinear group* $\mathsf{BG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G_1, G_2, \mathsf{e})$.
- $\mathsf{ES.KeyGen}(1^\ell) \rightarrow (\mathsf{sk}, \mathsf{pk})$*: a key generation algorithm that takes as input the size of messages in unary and outputs a key pair.*
- $\mathsf{ES.Sign}(\mathsf{sk}, M)$*: a signature algorithm that takes as input a secret key and a message array of length ℓ, and outputs a signature σ.*
- $\mathsf{ES.ChgRp}(\mathsf{pk}, M, \sigma, r)$*: a re-randomization algorithm that takes as input a public key, a message of length ℓ, a signature and a scalar $r \in \mathbb{Z}_p^*$, and returns a signature σ^* on message $M^* = r \cdot M$.*
- $\mathsf{ES.Verify}(\mathsf{pk}, M, \sigma) \rightarrow 0/1$*: a signature verification algorithm that takes as input a public key, a message of length ℓ and a signature, and outputs a bit encoding validity of the signature.*
- $\mathsf{ES.VKey}(\mathsf{sk}, \mathsf{pk}) \rightarrow 0/1$*: a key verification algorithm that takes as input a key pair and outputs a bit encoding consistency of the keys.*

We require an EQS scheme to satisfy correctness, existentially unforgeability under adaptive chosen-message attacks (EUF-CMA), *and* perfect signature adaption.

We define the security properties as follows:

Definition 4. *An EQS scheme with length parameter $\ell > 1$ is* correct *iff for all security parameters $\lambda \in \mathbb{N}$ and any bilinear group* $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$, *key pair* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{ES.KeyGen}(\mathsf{BG}, 1^\ell)$, *message $M \in (\mathbb{G}_1^*)^\ell$, and exponent $r \in \mathbb{Z}_p^*$ it holds that*

$$\Pr[\mathsf{ES.Verify}(\mathsf{pk}, M, \mathsf{ES.Sign}(\mathsf{sk}, M)) = 1] = 1 \quad \textit{and}$$
$$\Pr[\mathsf{ES.Verify}(\mathsf{pk}, rM, \mathsf{ES.ChgRp}(\mathsf{pk}, M, \mathsf{ES.Sign}(\mathsf{sk}, M), r)) = 1] = 1.$$

Definition 5. *An EQS scheme Π_{eqs} with length parameter $\ell > 1$ is* EUF-CMA *secure iff for all security parameters λ and PPT algorithms $\mathcal{A}$ there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\text{eqs-uf}}_{\Pi_{\mathrm{eqs}},\mathcal{A},\ell}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\text{eqs-uf}}_{\Pi_{\mathrm{eqs}},\mathcal{A},\ell}$ is defined as in Figure 1. We denote the maximal advantage of an adversary in winning the security game by $\mathsf{Adv}_{\text{eqs-uf}}$.

Definition 6. *An EQS system with length parameter $\ell > 1$ satisfies perfect signature adaption if for all security parameters λ and*

```
Experiment: $\mathsf{Exp}^{\mathsf{eqs\text{-}uf}}_{\Pi_{\mathsf{eqs}},\mathcal{A},\ell}(\lambda)$

(1) $\mathsf{pp} \leftarrow \mathsf{BGGen}(1^\lambda),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{ES.KeyGen}(1^\ell),\ Q \leftarrow \emptyset$
(2) $(M^*,\sigma^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sig}}}(\mathsf{pp},\mathsf{pk})$
(3) if $\exists (r,M) \in (\mathbb{Z}_p^* \times Q) : M^* = rM$ return 0
(4) if $\mathsf{ES.Verify}(\mathsf{pk}, M^*, \sigma^*) = 0$ return 0
(5) return 1

$O_{\mathsf{Sig}}(M)$ :
(5) $Q \leftarrow Q \cup \{M\}$
(6) return $\mathsf{ES.Sign}(\mathsf{sk}, M)$
```

Figure 1: EQS unforgeability (EUF-CMA) game.

```
Experiment: $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}(\lambda)$

(1) $Q \leftarrow \emptyset, (p,\mathbb{G}) \leftarrow \mathsf{GGen}(1^\lambda), G \leftarrow \mathbb{G}^*, (x_1,x_2,x_3) \overset{\$}{\leftarrow} \mathbb{Z}_p^3$
(2) $\mathsf{pk}_2 \leftarrow x_2G, \mathsf{pk}_3 \leftarrow x_3G,\ (y_2^*, y_3^*, (M_1^*, M_2^*)) \leftarrow \mathcal{A}^{O_{\mathsf{MAC}},O_{\mathsf{Vf}}}(\mathsf{pp},\mathsf{pk}_2,\mathsf{pk}_3)$
(3) if $(y_2^*, y_3^*) \in Q \vee M_1^* = 0 \vee M_2^* \neq (x_1 + x_2 \cdot y_2^* + x_3 \cdot y_3^*)M_1^*$
        return 0
    else return 1

$O_{\mathsf{MAC}}(y_2, y_3)$ :
(4) $Q \leftarrow Q \cup \{(y_2,y_3)\}, M_1 \overset{\$}{\leftarrow} \mathbb{G}^*, M_2 \leftarrow (x_1 + x_2 \cdot y_2 + x_3 \cdot y_3)M_1$
(5) return $(M_1, M_2)$

$O_{\mathsf{Vf}}(y_2, y_3, (M_1, M_2))$ :
(6) if $M_1 = 0 \vee M_2 \neq (x_1 + x_2 \cdot y_2 + x_3 \cdot y_3)M_1$ return 0 else return 1
```

Figure 2: MAC unforgeability (EUF-CMA) game.

each $\mathsf{pp} \leftarrow \mathsf{BGGen}(1^\lambda)$ *and tuple* $(\mathsf{sk},\mathsf{pk},M,\sigma,r)$ *with* $M \in (\mathbb{G}_1^*)^\ell$, $r \in \mathbb{Z}_p^*$, $\mathsf{ES.VKey}(\mathsf{pk},\mathsf{sk}) = 1$, *and* $\mathsf{ES.Verify}(\mathsf{pk},M,\sigma) = 1$, *it holds that the outputs of* $\mathsf{ES.ChgRp}(\mathsf{pk},M,\sigma,r)$ *and* $\mathsf{ES.Sign}(\mathsf{sk},rM)$ *are identically distributed.*

2.3 Algebraic MACs due to [12, 18]

One of our constructions employs the algebraic MAC scheme introduced by [18] and proven unforgeable under chosen message attacks including a verification oracle in the generic group model by [12]. In their construction the MAC keys are generated as $(x_1, x_2, x_3) \overset{\$}{\leftarrow} \mathbb{Z}_p^3$ and a MAC on a message $(y_2, y_3) \in \mathbb{Z}_p^2$ is computed as $(rG, (x_1 + x_2 \cdot y_2 + x_3 \cdot y_3) \cdot (rG))$ for $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. As we make use of the internal structure of the MAC scheme, we cannot use a generic MAC primitive in a black-box way, but make explicitly use of their construction. Therefore, we state the following theorem, which is straightforwardly implied by Theorem 2 of [12].

Theorem 1. *Let* GGen *be as above. For every* $\lambda \in \mathbb{N}$ *and* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{MAC}}$ *is defined as in Figure 2.*

3 Policy-based Anonymous Tokens with Private Metadata Bit and Public Metadata

We merge the definition of policy-based anonymous tokens [21] with that of anonymous tokens with private metadata bits and public metadata presented in [1, 4, 11, 27, 37]. While we follow the approach of [21] to model the policy as a list of allowed tokens, we omit the policy update from the formal definition. Instead, we treat the computation and distribution of the policy as an engineering aspect to be addressed by real-world deployments, analogous to maintaining the set of already spent tokens. Still, we note that, more complex policies can be realized on the application layer by implementing the policy via a predicate function checking validity of a tag or a tag derivation function computing a list of allowed tags for a given public system state. The system state could, for example, include the current date, enabling policies to activate new tags on a daily basis without requiring additional communication.

Definition 7. *A policy-based anonymous token scheme with private metadata bit and public metadata (*PB-MD-pbAT*) is a tuple of the following polynomial time algorithms:*

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$*: a probabilistic algorithm that takes as input the security parameter and returns the public parameters.*
- $\mathsf{SKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{sk},\mathsf{pk})$*: a probabilistic algorithm that takes as input the public parameters and generates a server key pair.*
- $\mathsf{Req}(\mathsf{pp},\mathsf{md},\mathsf{pk}) \rightarrow \mathsf{req}/\mathsf{state}$*: a probabilistic algorithm that takes as input the public parameters, some public metadata and the public key, and produces a pre-token request and an intermediate client state.*
- $\mathsf{Issue}(\mathsf{pp},\mathsf{req},\mathsf{md},\mathsf{ok},\mathsf{sk}) \rightarrow \mathsf{resp}/\bot$*: a probabilistic algorithm that takes as input the public parameters, the pre-token request, the metadata, a private bit and the server's secret key, and responds with a special error symbol or an issuance tuple that can be used by the client to generate a pre-token.*
- $\mathsf{Final}(\mathsf{pp},\mathsf{resp},\mathsf{state}) \rightarrow \rho/\bot$*: a deterministic algorithm that takes as input the public parameters, the server's issuance tuple and the client's intermediate state, and returns a pre-token or a special error symbol.*
- $\mathsf{Redeem}(\mathsf{pp},\rho,\tau) \rightarrow (\delta,\omega)$*: a probabilistic algorithm that takes as input the public parameters, a pre-token and a tag, and outputs a token-witness pair.*
- $\mathsf{Verify}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{pk}) \rightarrow \{0,1\}$*: a deterministic algorithm that takes as input the public parameters, a policy, a token, a tag, the public metadata, a witness and the server's public key, and returns a bit indicating acceptance of the token.*
- $\mathsf{ReadBit}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{sk}) \rightarrow \{\bot,0,1\}$*: a deterministic algorithm that takes as input the public parameters, a policy, a token, a tag, the public metadata, a witness and the server's secret key, and returns an error symbol if the token is not valid or the private bit associated with the pre-token.*

We require a PB-MD-pbAT *scheme to satisfy* correctness, freshness, one-more unforgeability, unlinkability *and* private bit privacy *which are defined as below.*

In a real-world deployment, the issuer executes $\mathsf{SKeyGen}(\cdot)$ and publishes pk. To request a pre-token, a client executes $\mathsf{Req}(\cdot)$, sends req to the issuer and stores state. Upon receiving req, the issuer executes $\mathsf{Issue}(\cdot)$ and returns resp to the client, which executes $\mathsf{Final}(\cdot)$ to derive the final pre-token. To redeem a token for a tag τ, the client executes $\mathsf{Redeem}(\cdot)$ and sends (δ, ω) to the respective verifier. The verifier checks that δ has not been received before and executes $\mathsf{Verify}(\cdot)$ to validate the token or $\mathsf{ReadBit}(\cdot)$ to read the private bit. The latter requires access to the secret issuance key.

3.1 Correctness

Correctness states that a correctly generated token verifies successfully and that the bit read from a correctly generated token is the one used in the issuance of the corresponding pre-token. Unlike [21], we allow a negligible failure probability. Alternatively, we could disallow metadata that hashes to 0. Furthermore, we require that the attempt to read a bit only fails if the token verification with the same parameters fails. This property formalizes the implicit assumption that the bit reading algorithm internally executes the token verification and returns $\perp$ only if the verification fails.

Definition 8. *A* PB-MD-pbAT *scheme is* correct *if for all security parameters* $\lambda \in \mathbb{N}$*, meta-data sizes* $\ell_{\mathsf{md}} = \ell_{\mathsf{md}}(\lambda)$*, tag sizes* $\ell_\tau = \ell_\tau(\lambda)$*, policy sizes* $\ell_{\mathcal{P}} = \ell_{\mathcal{P}}(\lambda)$*, public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, server keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SKeyGen}(\mathsf{pp})$*, policies* $\mathcal{P} \in \{0,1\}^{\ell_\tau \times \ell_{\mathcal{P}}}$*, tags* $\tau \in \mathcal{P}$*, metadata* $\mathsf{md} \in \{0,1\}^{\ell_{\mathsf{md}}}$*, and private bit* $\mathsf{ok} \in \{0,1\}$*, pre-tokens* $\rho \leftarrow \mathsf{Final}(\mathsf{pp}, \mathsf{resp}, \mathsf{state})$ *with* $(\mathsf{req}, \mathsf{state}) \leftarrow \mathsf{Req}(\mathsf{pp}, \mathsf{md}, \mathsf{pk})$ *and* $\mathsf{resp} \leftarrow \mathsf{Issue}(\mathsf{pp}, \mathsf{req}, \mathsf{md}, \mathsf{ok}, \mathsf{sk})$*, and each image* (δ, ω) *of* $\mathsf{Redeem}(\mathsf{pp}, \rho, \tau)$ *it holds that there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0] \leq \mathsf{negl}(\lambda) \text{ and}$$
$$\Pr[\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) \neq \mathsf{ok}] \leq \mathsf{negl}(\lambda),$$

and if for all $(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}, \mathsf{sk})$ *it holds that*

$$\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 1$$
$$\Leftrightarrow \mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) \neq \perp.$$

3.2 Freshness

In real-world deployments, verifiers prevent clients from spending the same token multiple times by checking, for each received token, whether it has already been redeemed. Consequently, we require a freshness property, which guarantees that tokens generated from different honestly produced pre-tokens or from different tags are distinct. Formally, we define freshness as follows:

Definition 9. *A* PB-MD-pbAT *scheme satisfies* freshness *if for all security parameters* $\lambda \in \mathbb{N}$*, meta-data sizes* $\ell_{\mathsf{md}} = \ell_{\mathsf{md}}(\lambda)$*, tag sizes* $\ell_\tau = \ell_\tau(\lambda)$*, policy sizes* $\ell_{\mathcal{P}} = \ell_{\mathcal{P}}(\lambda)$*, public parameters* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, server keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SKeyGen}(\mathsf{pp})$*, metadata* $\mathsf{md}_1, \mathsf{md}_2 \in \{0,1\}^{\ell_{\mathsf{md}}}$ *and private bits* $\mathsf{ok}_1, \mathsf{ok}_2 \in \{0,1\}$*. there exists a negligible function such that*

$$\Pr\left[\begin{array}{l} \delta_{0,0} = \delta_{0,1} \vee \delta_{0,0} = \delta_{1,0} \vee \delta_{0,0} = \delta_{1,1} : \\ \mathcal{P} \xleftarrow{\$} \{0,1\}^{\ell_\tau \times \ell_{\mathcal{P}}}, \tau_0 \xleftarrow{\$} \mathcal{P}, \tau_1 \xleftarrow{\$} (\mathcal{P} \setminus \{\tau_0\}), \\ (\mathsf{req}_b, \mathsf{state}_b) \leftarrow \mathsf{Req}(\mathsf{pp}, \mathsf{md}_b, \mathsf{pk}) \textit{ for } b \in \{0,1\}, \\ \mathsf{resp}_b \leftarrow \mathsf{Issue}(\mathsf{pp}, \mathsf{req}_b, \mathsf{md}_b, \mathsf{ok}_b, \mathsf{sk}) \textit{ for } b \in \{0,1\}, \\ \rho_b \leftarrow \mathsf{Final}(\mathsf{pp}, \mathsf{resp}_b, \mathsf{state}_b) \textit{ for } b \in \{0,1\}, \\ (\delta_{b,t}, \omega_{b,t}) \leftarrow \mathsf{Redeem}(\mathsf{pp}, \rho_b, \tau_t) \textit{ for } b, t \in \{0,1\} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

3.3 Unforgeability

We require that an adversary cannot produce more tokens for a given combination of public metadata and private bit than permitted by the policy for the number of pre-tokens issued for the given combination. This notion does not only require that an adversary cannot forge more tokens than permitted by the policy and the number of issued tokens, as done in [21], but also requires that a token generated from a pre-token issued for a combination of public metadata and private bit can only be verified with respect to the same combination. Formally, unforgeability is defined as follows:

Definition 10. *A* PB-MD-pbAT *scheme* Π *is* unforgeable *if for all security parameters* $\lambda \in \mathbb{N}$*, natural number* $n > 0$ *and* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A},n}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A}}$ *is defined as in Figure 3.*

3.4 Unlinkability

Unlinkability requires that the redemption of a token cannot be linked to the issuance of the token's pre-token or the redemption of another token derived from the same pre-token with a different tag. Formally, we define a security game, in which the adversary specifies a set of pre-tokens (via their indices) and a tag and receives a challenge containing a position k and a randomly permuted list of all tokens generated by the specified pre-tokens for the given tag. The adversary wins if it can find the index of the pre-token that has been used to generate the token at the k-th position after permutation. To ensure unlinkability between tokens generated by the same pre-token, we grant the adversary access to a redemption oracle, which provides the adversary with tokens generated with arbitrary pre-tokens (identified via an index) and tags. Obviously, the adversary is not allowed to query tokens that will end up in the challenge. This notion constitutes an extension of the unlinkability notion of [11].

Definition 11. *A* PB-MD-pbAT *scheme* Π *is* ϵ-unlinkable *if for all security parameters* $\lambda \in \mathbb{N}$*, integers* $n > 0$ *and* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}(\lambda) = 1] \leq \frac{\epsilon}{n} + \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}$ *is defined as in Figure 5.*

The bound $\frac{\epsilon}{n}$ stems from the fact that the issuer chooses the private bit embedded into the pre-tokens and observes the bit of received tokens. Consequently, the issuer can trivially reduce the anonymity set of a received token to all pre-token issuances executed with the private bit observed in that token. Consider an adversary in the security game that selects n_b pre-tokens with bit b for $b \in \{0,1\}$ and let $n = n_0 + n_1$ be the total number of selected pre-tokens. The adversary extracts bit b^* from the challenged token $(\delta_{\phi(k)})$ and then guesses at random among the indices where bit $b = b^*$ was used in the pre-token issuance. The adversary wins with probability $\frac{n_0}{n} \cdot \frac{1}{n_0} + \frac{n_1}{n} \cdot \frac{1}{n_1} = \frac{2}{n}$. We require that guessing a random pre-token in the set of pre-tokens issued with the private bit embedded into the challenged token to be the best strategy and, hence, focus on 2-unlinkability in this work.

```
Experiment: $\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A},n}(\lambda)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{SKeyGen}(\mathsf{pp})$
(2) $(\mathcal{P}, \mathsf{ok}^*, \mathsf{md}^*, \{(\delta_i,\tau_i,\omega_i)\}_{i\in[n]}) \leftarrow \mathcal{A}^{O_{\mathsf{Is}},O_{\mathsf{RB}}}(\mathsf{pp},\mathsf{pk})$
(3) if $\exists i \in [n]$ s.t. $\mathsf{ReadBit}(\mathsf{pp},\mathcal{P},\delta_i,\tau_i,\mathsf{md},\omega_i,\mathsf{sk}) \neq \mathsf{ok}^*$ then return 0
(4) if $|\mathcal{P}| \cdot i_{\mathsf{ok}^*,\mathsf{md}^*} \geq n$ or $\exists i \neq j$ s.t. $\delta_i = \delta_j$ then return 0 else return 1

$O_{\mathsf{Is}}(\mathsf{req},\mathsf{md},\mathsf{ok})$ :
(5) if $i_{\mathsf{ok},\mathsf{md}} = \mathbf{null}$, then $i_{\mathsf{ok},\mathsf{md}} \leftarrow 1$ else $i_{\mathsf{ok},\mathsf{md}} \leftarrow i_{\mathsf{ok},\mathsf{md}} + 1$
(6) return $\mathsf{resp} \leftarrow \mathsf{Issue}(\mathsf{pp},\mathsf{req},\mathsf{md},\mathsf{ok},\mathsf{sk})$

$O_{\mathsf{RB}}(\mathcal{P},\delta,\tau,\mathsf{md},\omega)$ :
(5) return $\mathsf{ReadBit}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{sk})$
```

Figure 3: Unforgeability security game.

```
Experiment: $\mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}(\lambda)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j \leftarrow 0,\ Q_{\mathsf{redeem}} \leftarrow \emptyset$
(2) $(\mathsf{pk}, s_0) \leftarrow \mathcal{A}(0,\mathsf{pp}),\ (Q,\tau,s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}},O_{\mathsf{Is}},O_{\mathsf{Rd}}}(1,s_0)$
(3) if $|Q| < n \vee \exists (i,i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ return 0
(4) if $\exists i \in Q$ s.t. $((i,\tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null},\bot\})$ return 0
(5) if $Q$ contains duplicates return 0
(6) $\forall i \in Q : (\delta_i,\cdot,\omega_i) \leftarrow \mathsf{Redeem}(\mathsf{pp},\rho_i,\tau)$
(7) $k \xleftarrow{\$} Q$, pick a random permutation $\phi$ of $Q$
(8) $k^* \leftarrow \mathcal{A}(2,s_1,\phi(k),\{\delta_{\phi(i)},\omega_{\phi(i)}\}_{i\in Q})$
(9) if $k = k^*$ return 1 else return 0

$O_{\mathsf{Req}}(j,\mathsf{md})$ :
(10) $j \leftarrow j+1, \mathsf{md}_j \leftarrow \mathsf{md}, (\mathsf{req}_j,\mathsf{state}_j) \leftarrow \mathsf{Req}(\mathsf{pp},\mathsf{md},\mathsf{pk})$
(11) return $\mathsf{req}_j$

$O_{\mathsf{Is}}(i,\mathsf{resp})$ :
(12) if $\mathsf{state}_i = \mathbf{null} \vee \rho_i \neq \mathbf{null}$ return $\bot$
(13) $\rho_i \leftarrow \mathsf{Final}(\mathsf{pp},\mathsf{resp},\mathsf{state}_i)$
(14) if $\rho_i = \bot$ return 0 else return 1

$O_{\mathsf{Rd}}(i,\tau)$ :
(15) if $\rho_i \in \{\mathbf{null},\bot\} \vee (i,\tau) \in Q_{\mathsf{redeem}}$ return $\bot$
(16) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i,\tau)\}$
(17) return $\mathsf{Redeem}(\mathsf{pp},\rho_i,\tau)$
```

Figure 4: Unlinkability security game.

3.5 Private Bit Privacy

We require that an adversary cannot distinguish the private metadata bit embedded into a pre-token. To formalize this property, we adapt the definition of [11] to the setting of policy-based tokens. Most importantly, we issue pre-tokens rather than plain tokens and remove the token verification oracle, since tokens are publicly verifiable via $\mathsf{Verify}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{pk})$.

Definition 12. *A* PB-MD-pbAT *scheme* Π *is* private bit private *if for all security parameters* $\lambda \in \mathbb{N}$ *and* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ *is defined as in Figure 5.*

```
Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{SKeyGen}(\mathsf{pp}), c \leftarrow 0$
(2) return $\mathcal{A}^{O_{\mathsf{Is}},O_{\mathsf{Ch}},O_{\mathsf{RB}}}(\mathsf{pp},\mathsf{pk})$

$O_{\mathsf{Is}}(\mathsf{ok},\mathsf{md},\mathsf{req})$ :
(1) return $\mathsf{Issue}(\mathsf{pp},\mathsf{req},\mathsf{md},\mathsf{ok},\mathsf{sk})$

$O_{\mathsf{Ch}}(\mathsf{md},\mathsf{req})$ :
(1) if $c = 1$ return $\bot$
(2) $c \leftarrow 1,\ \mathsf{md}^* \leftarrow \mathsf{md}$
(3) return $\mathsf{Issue}(\mathsf{pp},\mathsf{req},\mathsf{md},\mathsf{ok}^*,\mathsf{sk})$

$O_{\mathsf{RB}}(\mathcal{P},\delta,\tau,\mathsf{md},\omega)$ :
(1) if $c = 1 \wedge \mathsf{md} = \mathsf{md}^*$ return $\bot$
(2) return $\mathsf{ReadBit}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{sk})$
```

Figure 5: Private bit privacy security game.

3.6 Without Public Verifiability

Our definition requires tokens to be publicly verifiable, i.e., requires the existence of a verification algorithm Verify that validates tokens based on the issuer's public key, without knowledge of the secret issuance key. However, we also present a construction that sacrifices public verifiability in exchange for improved efficiency. To adapt the definition of policy-based anonymous tokens with metadata to privately verifiable policy-based anonymous tokens with metadata, one simply removes the verification algorithm Verify, replaces all occurrences of $\mathsf{Verify}(\ldots,\mathsf{pk}) = 1$ in the definitions by $\mathsf{ReadBit}(\ldots,\mathsf{sk}) \neq \bot$ and occurrences of $\mathsf{Verify}(\ldots,\mathsf{pk}) = 0$ by $\mathsf{ReadBit}(\ldots,\mathsf{sk}) = \bot$ and augments the private bit privacy security games with a verification oracle that receives $(\mathcal{P},\delta,\tau,\mathsf{md},\omega)$ and returns 0 if $\mathsf{ReadBit}(\mathsf{pp},\mathcal{P},\delta,\tau,\mathsf{md},\omega,\mathsf{sk}) = \bot$ and 1 otherwise.

4 Publicly Verifiable Policy-based Tokens with Metadata from Equivalence Class Signatures

Since we provide an extensive Technical Overview (c.f. Section1.2), we will focus on the technical presentation of our construction in the following. Furthermore, we present additional extensions, i.e., for non-interactiveness, non-transferability and hidden tags in Appendix A.

4.1 The Construction

We present our construction in Figure 6. The core idea is to encode the pre-token's relevant information, the client's key sk_c, the metadata md, and the private bit ok in a message $(R, \mathsf{sk}_c R, \mathcal{H}_1(\mathsf{md}) \cdot R, \mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}} R)$ that is signed via an equivalence class signature scheme. For redemption, the client re-randomizes the signature and the message. While the re-randomization prevents the server from linking the re-randomized signature or re-randomized message (R^*, M_1, M_2, M_3) to the original, it ensures that the discrete logarithm of M_1, M_2, and M_3 to R^* remains constant. Besides binding the message to the metadata and the private bit, this also allows us to authenticate the token $\delta = \mathsf{sk}_c \cdot \mathcal{H}_2(\tau)$ by proving that $\mathsf{dlog}_{R^*}(M_1) = \mathsf{dlog}_{\mathcal{H}_2(\tau)}(\delta)$.

We note that the construction can straightforwardly be adapted to remove support for the public metadata and the private bit. This improves both communication and computation complexity. We provide more details of the influence of the individual properties on the construction's complexity in Section 6. Interestingly, the zero-knowledge proof of knowledge of sk_c created by the client as part of its request is only required for the private bit privacy property. A scheme without a private metadata bit can be instantiated without the request proof.

Finally, we note that clients can remove the variable X^* from the witness if they know that the verifier is in possession of the secret issuance key, e.g., if the verifier can also read the private bit. A verifier in possession of the secret issuance key can compute the two possible X^* candidates based on R^* and perform two verifications, one for each candidate.

4.2 Zero-knowledge Proofs

Our construction utilizes four different zero-knowledge proof of knowledge systems. We introduce the proof systems in the following. If the underlying equivalence class signature scheme is instantiated according to [22], all proof systems can be instantiated with generalized Schnorr proofs [34] as detailed in Appendix D.

The server public key pk contains a zero-knowledge proof π_{pk} ensuring the clients that the equivalence class signature public key $\mathsf{pk}_{\mathsf{eqs}}$ has been generated correctly or, to be more precise, that the server knows the secret key $\mathsf{sk}_{\mathsf{eqs}}$ corresponding to $\mathsf{pk}_{\mathsf{eqs}}$. We define the following proof relation

$$\mathsf{ZKP}_{\mathsf{eqs}} = \mathsf{ZKP}\Big\{\mathsf{sk}_{\mathsf{eqs}} : \mathsf{ES.VKey}(\mathsf{pp}, \mathsf{sk}_{\mathsf{eqs}}, \mathsf{pk}_{\mathsf{eqs}}) = 1\Big\}$$

and assume $(\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify})$ to be a zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{eqs}}$.

To request a token token, the client sends a public key $\mathsf{pk}_c = \mathsf{sk}_c G_1$ to the server and proves in zero-knowledge that it knows the corresponding secret key sk_c. For this proof system, we define relation

$$\mathsf{ZKP}_{\mathsf{dlog}} = \mathsf{ZKP}\Big\{\mathsf{sk}_c : \mathsf{pk}_c = \mathsf{sk}_c G_1\Big\}$$

and assume $(\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify})$ to be a secure zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{dlog}}$.

If a malicious server is able to embed arbitrary private information into the pre-token, it can use the information to break unlinkability of tokens. Hence, we have to ensure that the private information is just a single bit. Therefore, we require the server to proof that it used one of the two public private bit labels $\mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}$ to generate a pre-token. Formally, we define the following proof relation

$$\mathsf{ZKP}_{\mathsf{eq\text{-}or}} = \mathsf{ZKP}\Big\{(v, b) : R = vG_1 \wedge X = v \cdot \mathsf{pk}^b_{\mathsf{pb}}\Big\}$$

and assume $(\mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify})$ to be a secure zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{eq\text{-}or}}$.

Finally, the client has to prove that the pre-token's secret key sk_c used for the token computation is the one that is authenticated by the equivalence class signatures, i.e., equals the discrete logarithm of the re-randomized message's second component pk^*_c to the basis R^*, which is the first component. We define the following relation

$$\mathsf{ZKP}_{\mathsf{eq}} = \mathsf{ZKP}\Big\{\mathsf{sk}_c : \delta = \mathsf{sk}_c T \wedge \mathsf{pk}^*_c = \mathsf{sk}_c R^*\Big\}$$

Construction 1: EQS-based PB-MD-pbAT

```
Setup(1^λ) :
(1)  Output $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G_1, G_2, e) \leftarrow \mathsf{BGGen}(1^\lambda)$.
SKeyGen(pp) :
(2)  Compute $(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{pk}_{\mathsf{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$.
(3)  Compute $\pi_{\mathsf{pk}} \leftarrow \mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Prove}(\mathsf{sk}_{\mathsf{eqs}}, (\mathsf{pp}, \mathsf{pk}_{\mathsf{eqs}}))$.
(4)  Sample $\mathsf{sk}^b_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathsf{pk}^b_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} G_1$ for $b \in \{0,1\}$.
(5)  Define $\mathsf{pk} \leftarrow (\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
     and $\mathsf{sk} \leftarrow (\mathsf{sk}_{\mathsf{eqs}}, \mathsf{sk}^0_{\mathsf{pb}}, \mathsf{sk}^1_{\mathsf{pb}}, \mathsf{pk})$.
(6)  Output $(\mathsf{sk}, \mathsf{pk})$.
Req(pp, md, pk) :
(7)  If $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, (\mathsf{pp}, \mathsf{pk}_{\mathsf{eqs}})) = 0$, output $(\bot, \bot)$.
     (This step is only executed once per server public key.)
(8)  Sample $\mathsf{sk}_c \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathsf{pk}_c \leftarrow \mathsf{sk}_c G_1$.
(9)  Compute $\pi_{\mathsf{req}} \leftarrow \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}(\mathsf{sk}_c, (G_1, \mathsf{pk}_c))$.
(10) Output $\mathsf{req} = (\mathsf{pk}_c, \pi_{\mathsf{req}})$ and $\mathsf{state} = (\mathsf{sk}_c, \mathsf{md}, \mathsf{pk})$.
Issue(pp, req, md, ok, sk) :
(11) If $\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$, output $\bot$.
(12) Sample $v \xleftarrow{\$} \mathbb{Z}^*_p$ and define $R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c$ and $X \leftarrow v \cdot \mathsf{pk}^{\mathsf{ok}}_{\mathsf{pb}}$.
(13) Compute $\pi_{\mathsf{is}} \leftarrow \mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Prove}((v, \mathsf{ok}), (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}))$.
(14) Compute $m \leftarrow \mathcal{H}_1(\mathsf{md})$ and define $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$.
(15) Compute $\sigma \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg})$.
(16) Output $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
Final(pp, resp, state) :
(17) If $\mathsf{state} = \bot$, output $\bot$.
(18) Compute $m \leftarrow \mathcal{H}_1(\mathsf{md})$ and define $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$.
(19) If $\mathsf{pk}'_c \neq \mathsf{sk}_c R$, output $\bot$.
(20) If $\mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify}(\pi_{\mathsf{is}}, (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})) = 0$, output $\bot$.
(21) If $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma) = 0$, output $\bot$.
(22) Output $\rho = (\mathsf{msg}, \mathsf{sk}_c, \sigma)$.
Redeem(pp, ρ, τ) :
(23) Sample $r \xleftarrow{\$} \mathbb{Z}^*_p$ and compute $\mathsf{msg}^* = r \cdot \mathsf{msg}$ and $\delta \leftarrow \mathsf{sk}_c \mathcal{H}_2(\tau)$.
(24) Let $(R^*, \mathsf{pk}^*_c, M^*, X^*) = \mathsf{msg}^*$.
(25) Compute $\sigma^* \leftarrow \mathsf{ES.ChgRp}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma, r)$.
(26) Compute $\pi_{\mathsf{red}} \leftarrow \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}((\mathsf{sk}_c), (\mathsf{pk}^*_c, \delta, R^*, \mathcal{H}_2(\tau))$.
(27) Output $(\delta, \tau, \omega = (R^*, \mathsf{pk}^*_c, X^*, \sigma^*, \pi_{\mathsf{red}}))$.
Verify(pp, P, δ, τ, md, ω, pk) :
(28) If $\tau \notin \mathcal{P}$, output 0.
(29) Define $\mathsf{msg} \leftarrow (R^*, \mathsf{pk}^*_c, \mathcal{H}_1(\mathsf{md}) \cdot R^*, X^*)$.
(30) If $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma^*) = 0$, output 0.
(31) Output $\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify}(\pi_{\mathsf{red}}, (\mathsf{pk}^*_c, \delta, R^*, \mathcal{H}_2(\tau))$.
ReadBit(pp, P, δ, τ, md, ω, sk) :
(32) If $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0$, output $\bot$.
(33) If $X^* = \mathsf{sk}^1_{\mathsf{pb}} R^*$, output 1. Otherwise, output 0.
```

Figure 6: PB-MD-pbAT **construction based on equivalence class signatures.**

and assume $(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$ to be a zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{eq}}$.

4.3 Security

With regard to security, we state the following theorem:

Theorem 2. *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be random oracles, let* ES *be a secure equivalence class signature scheme as defined in Definitions 3-6 and let* $(\mathsf{ZKP}_{\mathsf{rel}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{rel}}.\mathsf{Verify})$ *be a secure zero-knowledge proof of knowledge for relation* $\mathsf{ZKP}_{\mathsf{rel}}$ *according to Definition 1 for* $\mathsf{rel} \in \{\mathsf{eqs}, \mathsf{dlog}, \mathsf{eq\text{-}or}, \mathsf{eq}\}$. *Assume that the XDH assumption holds for* BGGen. *Then, the construction described in Construction 1 (Figure 6) is a secure* PB-MD-pbAT *scheme according to Definitions 7-12.*

We present a detailed security proof determining the concrete security of the construction in Appendix B.

5 Privately Verifiable Policy-based Tokens from Algebraic MACs

In this section, we present our policy-based token scheme with metadata based on the algebraic MACs construction of [18]. This construction sacrifices public verifiability of tokens to increase the efficiency, both in terms of communication and computation.

5.1 The Construction

The construction is presented in Figure 7. We encode the pre-token's information in a message $(\mathsf{sk}_c, \mathcal{H}_1(\mathsf{md}))$ that is authenticated via an algebraic MAC $(M_1, M_2) = (vG, (\mathsf{sk}^s_{b,1} + \mathsf{sk}^s_{b,2} \cdot \mathsf{sk}_c + \mathsf{sk}^s_{b,3} \cdot \mathcal{H}_1(\mathsf{md})) \cdot vG)$. To encode the private bit, we make use of two different MAC keys, one for each bit b. As the client requesting a pre-token cannot send sk_c in clear, it sends a public key $\mathsf{pk}_c = \mathsf{sk}_c G$ and proves that it knows $\mathsf{dlog}_G(\mathsf{pk}_c)$. The MAC is then computed based on pk_c; note that $(\mathsf{sk}^s_{b,2} \cdot \mathsf{sk}_c) \cdot vG = (\mathsf{sk}^s_{b,2} \cdot v)\mathsf{pk}_c$. To prevent malicious issuers from issuing inconsistent MACs, e.g., to break the unlinkability guarantees, we require the issuer to proof correct issuance of the MACs in zero-knowledge. However, in order to proof that a MAC has been generated correctly, it is necessary for the issuer to publish some kind of public key that can be used as a reference for binding the issuer to its MAC keys. For the second and third MAC key component, the issuer publishes public keys $\mathsf{pk}^s_{b,i} = \mathsf{sk}^s_{b,i}G$ for $b\{0,1\}$ and $i \in \{2,3\}$. Unfortunately, the MAC scheme does not allow publishing $\mathsf{pk}^s_{b,1} = \mathsf{sk}^s_{b,1}G$. Hence, we include a Pedersen [32] commitment $C_b = u_b G \mathsf{sk}^s_{b,1} H$ for $u_b \xleftarrow{\$} \mathbb{Z}_p$ on the first MAC key component into the public key. When issuing a pre-token, the issuer proves that the MAC has been created correctly with respect to one of the secret MAC keys committed to via the Pedersen commitment C_b and public keys $(\mathsf{pk}^s_{b,2}, \mathsf{pk}^s_{b,3})$ without disclosing the bit b.

For the redemption, the client re-randomized the MAC to $(M^*_1, M^*_2) = (rM_1, rM_2)$ which authenticates the randomized client public key $\mathsf{pk}'_c = \mathsf{sk}_c(rM^*_1)$. With the re-randomized public key, the client authenticates the token $\delta = \mathsf{sk}_c \cdot \mathcal{H}_2(\tau)$ by proving that $\mathsf{dlog}_{\mathcal{H}_2(\tau)}(\delta) = \mathsf{dlog}_{M^*_1}(\mathsf{pk}'_c)$.

5.2 Zero-knowledge Proofs

In addition to the zero knowledge proof systems $(\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify})$ and $(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$ introduced above, we make use of two additional proof systems. Both proof systems can be efficiently instantiated with generalized Schnorr proofs as illustrated in Appendix D.

For the public keys $\{\mathsf{pk}^s_{b,i}\}_{b\in\{0,1\}, i\in\{2,3\}}$, the issuer publishes a zero-knowledge proof of knowledge of the corresponding secret keys. Formally, we define the following relation

$$\mathsf{ZKP}_{\mathsf{4DL}} = \mathsf{ZKP}\Big\{(\{\mathsf{sk}^s_i\}_{i\in\{2,3\}}) : \mathsf{pk}^s_i = \mathsf{sk}^s_i G \text{ for } i \in \{2,3\}\Big\}$$

and assume $(\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify})$ to be a secure zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{4DL}}$.

When issuing a pre-token, the issuer has to prove correct creation of the MAC. The corresponding proof relation is defined as

$$\mathsf{ZKP}_{\mathsf{PdOr}} = \mathsf{ZKP}\Big\{(\mathsf{sk}^s_{b,1}, u_b, v, b) : C_b = u_b G + \mathsf{sk}^s_b H$$
$$\wedge\, M_1 = vG \wedge M_2 = \mathsf{sk}^s_{b,1} M_1 + v K_b\Big\}.$$

where K_b is computed by the prover as $K_b = \mathcal{H}_1(\mathsf{md}) \cdot \mathsf{pk}^s_{b,3} + \mathsf{sk}^s_{b,2} \cdot \mathsf{pk}_c$ and by the verifier as $K_b = \mathcal{H}_1(\mathsf{md}) \cdot \mathsf{pk}^s_{b,3} + \mathsf{sk}_c \cdot \mathsf{pk}^s_{b,2}$. We assume $(\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Verify})$ to be a secure zero-knowledge proof of knowledge for relation $\mathsf{ZKP}_{\mathsf{PdOr}}$.

5.3 Security

With regard to security, we state the following theorem:

Theorem 3. *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be random oracles, let* $(\mathsf{ZKP}_{\mathsf{rel}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{rel}}.\mathsf{Verify})$ *be secure zero-knowledge proofs of knowledge for relation* $\mathsf{ZKP}_{\mathsf{rel}}$ *according to Definition 1 for* $\mathsf{rel} \in \{\mathsf{4DL}, \mathsf{dlog}, \mathsf{PdOr}, \mathsf{eq}\}$, *and let* $(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$ *satisfy statement-oblivious knowledge soundness according to Definition 2. Assume Theorem 1 and the DDH assumption to hold for group generation algorithm* GGen. *Then, the construction described in Construction 2 (Figure 7) is a secure privately verifiable* PB-MD-pbAT *scheme according to the adaption of Definitions 7-12 described in Section 3.6.*

We provide a proof draft in Appendix C.

6 Evaluation

We have implemented the three constructions that are part of our framework, all with and without the private bit and public metadata property, executed benchmarks and analyzed the communication complexity. For comparison, we have also implemented the policy-based anonymous token scheme of [21] and the state-of-the-art standard anonymous token scheme Privacy Pass [17]. As the protocol of [17] incorporates additional features going beyond the scope of anonymous tokens, e.g., the identification of the requested resource via some requested binding data, we trim the Privacy Pass protocol to the plain anonymous token scheme to ensure a fair comparison. Most importantly, we remove the token-based HMAC computation in the redemption phase and simply send the token received during the token issuance in clear.

We instantiate the equivalence class signature scheme in the implementation of our EQS-based construction and the implementation of [21] with the scheme proposed by [22]. For the signature scheme and the zkSNARK in our zkSNARK-based construction, we make use of Schnorr signatures [34] and the Groth16 [24] proof system. The advantage of the Groth16 proof system is its small proof size that is independent of the proof relation's complexity. The major downside of Groth16 is the toxic waste produced during the generation of the proof system's parameters that can be used to forge proofs for statements without the knowledge of the corresponding witness. However, in our setting, we have a central authority for issuing tokens, the token issuer, that, hence, can be

Construction 2: MAC-based PB-MD-pbAT

```
Setup(1^λ):
(1) Sample $(\mathbb{G}, p, G) \leftarrow \mathsf{GGen}(1^\lambda)$ and $H \xleftarrow{\$} \mathbb{G}^*$ and output $\mathsf{pp} = (\mathbb{G}, p, G, H)$.

SKeyGen(pp):
(2) Sample $\mathsf{sk}^s_{b,i}, u_b \xleftarrow{\$} \mathbb{Z}_p$ for $b \in \{0,1\}$ and $i \in [3]$.
(3) Compute $\mathsf{pk}^s_{b,i} \leftarrow \mathsf{sk}^s_{b,i} G$ for $b \in \{0,1\}$ and $i \in \{2,3\}$,
    $C_b \leftarrow u_b G + \mathsf{sk}^s_b H$ for $b \in \{0,1\}$
    and $\pi_{\mathsf{pk}} \leftarrow \mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}(\{\mathsf{sk}^s_{b,i}\}_{b\in\{0,1\},i\in\{2,3\}}, (G, \{\mathsf{pk}^s_{b,i}\}_{b\in\{0,1\},i\in\{2,3\}})$.
(4) Define $\mathsf{pk} = (\{C_b, \{\mathsf{pk}^s_{b,i}\}_{i\in\{2,3\}}\}_{b\in\{0,1\}}, \pi_{\mathsf{pk}})$
    and $\mathsf{sk} = (\{u_b, \{\mathsf{sk}^s_{b,i}\}_{i\in[3]}\}_{b\in\{0,1\}}, \mathsf{pk})$ and output $(\mathsf{sk}, \mathsf{pk})$.

Req(pp, md, pk):
(5) If $\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, (G, \{\mathsf{pk}^s_{b,i}\}_{b\in\{0,1\},i\in\{2,3\}})) = 0$, output $(\perp, \perp)$
    (This step is only executed once per server public key.)
(6) Sample $\mathsf{sk}_c \xleftarrow{\$} \mathbb{Z}_p$ and compute $\mathsf{pk}_c \leftarrow \mathsf{sk}_c G$
    and $K_b \leftarrow \mathsf{sk}_c \mathsf{pk}^s_{b,2} + \mathcal{H}_1(\mathsf{md}) \cdot \mathsf{pk}^s_{b,3}$ for $b \in \{0,1\}$.
(7) Compute $\pi_{\mathsf{req}} \leftarrow \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}(\mathsf{sk}_c, (G, \mathsf{pk}_c))$.
(8) Output $(\mathsf{req} = (\mathsf{pk}_c, \pi_{\mathsf{req}}), \mathsf{state} = (\mathsf{sk}_c, \mathsf{pk}, K_0, K_1))$.

Issue(pp, req, md, ok, sk):
(9) If $\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify}(\pi_{\mathsf{req}}, (G, \mathsf{pk}_c)) = 0$, output $\perp$.
(10) Sample $v \xleftarrow{\$} \mathbb{Z}^*_p$ and compute
    $K_b \leftarrow \mathsf{sk}^s_{\mathsf{ok},2} \cdot \mathsf{pk}_c + \mathcal{H}_1(\mathsf{md}) \cdot \mathsf{pk}^s_{b,3}$ for $b \in \{0,1\}$,
    $M_1 \leftarrow vG$, $M_2 \leftarrow \mathsf{sk}^s_{\mathsf{ok},1} M_1 + vK_{\mathsf{ok}}$, and
    $\pi_{\mathsf{is}} \leftarrow \mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Prove}((\mathsf{sk}^s_{\mathsf{ok},1}, u_{\mathsf{ok}}, v, \mathsf{ok}), (G, H, M_1, M_2, \{C_b, K_b\}_{b\in\{0,1\}}))$.
(11) Output $\mathsf{resp} = (M_1, M_2, \pi_{\mathsf{is}})$.

Final(pp, resp, state):
(12) If $\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Verify}(\pi_{\mathsf{is}}, (G, H, M_1, M_2, \{C_b, K_b\}_{b\in\{0,1\}})) = 0$
    or $\mathsf{state} = \perp$, output $\perp$.
(13) Output $\rho = (\mathsf{sk}_c, M_1, M_2)$.

Redeem(pp, ρ, τ):
(14) Sample $r \xleftarrow{\$} \mathbb{Z}^*_p$ and compute $M^*_1 \leftarrow rM_1$, $M^*_2 \leftarrow rM_2$, $\mathsf{pk}^*_c \leftarrow \mathsf{sk}_c M^*_1$
    and $\delta \leftarrow \mathsf{sk}_c \mathcal{H}(\tau)$.
(15) Compute $\pi_{\mathsf{red}} \leftarrow \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}(\mathsf{sk}_c, (\mathsf{pk}^*_c, \delta, M^*_1, \mathcal{H}_2(\tau))$.
(16) Output $(\delta, \tau, \omega = (M^*_1, M^*_2, \pi_{\mathsf{red}}))$.

ReadBit(pp, 𝒫, δ, τ, md, ω, pk):
(17) If $\tau \notin \mathcal{P}$ or $M^*_1 = 0$, output $\perp$.
(18) For $b \in \{0,1\}$, compute
    $\mathsf{pk}^*_b \leftarrow \frac{1}{\mathsf{sk}^s_{b,2}}(M^*_2 - (\mathsf{sk}^s_{b,1} + \mathsf{sk}^s_{b,3} \cdot \mathcal{H}_1(\mathsf{md})) \cdot M^*_1)$
    and $\mathsf{ok}_b \leftarrow \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify}(\pi_{\mathsf{red}}, (\mathsf{pk}^*_b, \delta, M^*_1, \mathcal{H}(\tau))$.
(19) Output $\perp$ if $\mathsf{ok}_0 = \mathsf{ok}_1 = 0$, 1 if $\mathsf{ok}_1 = 1$ and 0 otherwise.
```

Figure 7: PB-MD-pbAT **construction based on algebraic MACs.**

trusted with the generation of the zkSNARK parameters – leaking the toxic waste is equivalent to leaking the issuance keys, which is against the issuer's own security interests.

6.1 Benchmarks

Experimental setup. We have implemented the constructions and benchmarks in Rust (compiled with rustc 1.84.0 (9fc6b4312)) and executed all our benchmarks on a Laptop with an Intel(R) Core(TM) i7-1360P @ 2.20 GHz processor and 16GB of RAM. All benchmarks are repeated 100 times. We measure the execution time of server and client in the request phase and the redemption phase individually. As the client's request submission and finalization are executed non-continuously, we measure these steps individually, sum them up, and report them as *request* runtime. Our experiments do not consider network delays. The overhead of message transmission can be incorporated based on the respective network setting and the communication complexity reported in Section 6.2.

We benchmark each of our constructions without any metadata, with the private metadata bit property, with the public metadata property, and with both properties. In the constructions with the private metadata bit property, we do not differentiate between Verify and ReadBit and report the runtime of ReadBit as the token verification runtime. As Privacy Pass supports a batched token issuance, we bench the token issuance for batches of $N \in \{1, 10, 25, 50, 100\}$ tokens. Since the batched issuance does not affect the redemption, we only consider redemptions of single tokens.

We use the *BLS12-381* curve for the implementations of our EQS-based construction and the construction of [21], and the Bandersnatch [29] curve for the implementations of the MAC-based construction and Privacy Pass [17]. In the implementation of the zkSNARK-based construction, we utilize the Bandersnatch curve for the protocol implementation and base the zkSNARK on the *BLS12-381* curve. For all curve operations as well as the zkSNARK computation and verification, we utilize the Arkworks framework [15].

Results. We display the results of our benchmarks in Figures 8, 9, and 10. In Figure 8, we compare the runtime of our EQS-based and MAC-based constructions with Privacy Pass [17] (with batch sizes of 1 and 10) and the construction of Faut et al. [21]. The runtime of the Privacy Pass request and issuance increases approximately linearly as the batch size increases, i.e., the request runtime and the issuance runtime in ms increase to $(9.8, 19, 41.1)$ and $(6, 13.4, 26.3)$ for batches of $N = (25, 50, 100)$. In Figure 9, we report the runtime of our zkSNARK-based instantiation. Its runtime is comparable to the other constructions in all steps but the client's token redemption. During the token redemption, the client creates a zkSNARK, which requires roughly two orders of magnitude more time than the redemption in the other constructions. Finally, we display the client and server initialization runtime of all constructions in Figure 10. In the EQS-based and MAC-based construction, we move the verification of the issuer keys during the request algorithm into a dedicated client-side initialization as the verification needs to be performed just once. Furthermore, we assign the task of creating the zkSNARK parameters in our zkSNARK-based construction to the issuer as part of the issuance key generation. In some constructions, clients do not need a dedicated initialization, i.e., if they do not need to verify the server's public key, which is why we include only some of the constructions in the client initialization part.

6.2 Communication Complexity

In Table 1, we report the communication complexity of the plain versions of all our constructions as well as the additional overhead introduced by including the private metadata bit or the public metadata property. For comparison, we also include the communication complexity of Privacy Pass [17] and Faut et al. [21]. We encode points in $\mathbb{G}$ on the Bandersnatch curve with 32 bytes, points in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on the BLS12-381 curve with 48, 96 and 576 bytes, and scalars in both curves with 32 bytes. The Groth16 zkSNARK

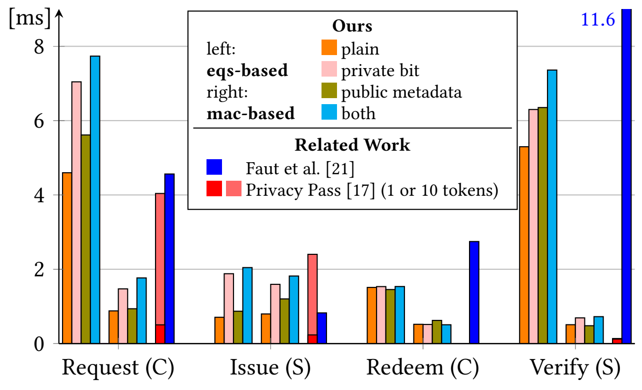

Figure 8: Runtime of our EQS-based and MAC-based constructions with and without private metadata bits and public metadata and our implementations of [17] and [21]. For [17], we also report the cost for issuing a batch of 10 tokens at once. As batching has not effect on the redemption, we only report this number for the issuance.

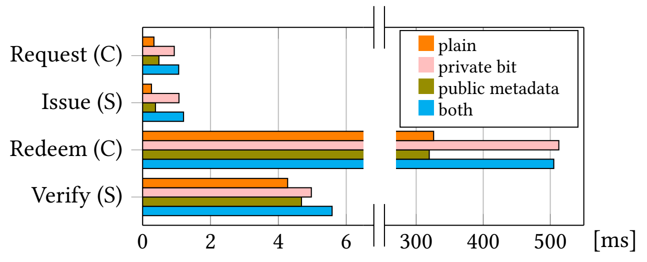

Figure 9: Runtime of our zkSNARK-based construction with and without private metadata bits and public metadata. The zkSNARK circuits consist of 11 582, 19 298, 11 969, and 19 685 constraints (plain, private-bit, public-metadata, both).

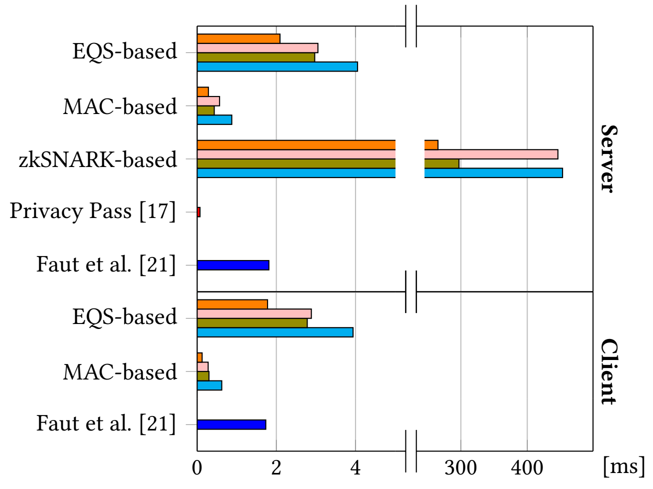

Figure 10: Runtime of the server initialization and the client initialization (if required) for all constructions.

Table 1: Communication complexity in bytes of our constructions, Privacy Pass [17], and Faut et al. [21] as well as the overhead of adding the private metadata bit and the public metadata property to our constructions.

	Issuance		Redemption
	Client	Server	Client
Privacy Pass [17]	32 B	96 B	64 B
+ each extra token	32 B	32 B	64 B
Faut et al. [21]	96 B[2]	192 B	$\log_{256}(\ell_{\mathcal{P}})$ + 960 B
EQS-based	48 B	288 B	$\log_{256}(\ell_{\mathcal{P}})$ + 400 B
+ private bit	64 B	176 B	48 B
+ public metadata	0 B	0 B	0 B
MAC-based	96 B	192 B	$\log_{256}(\ell_{\mathcal{P}})$ + 160 B
+ private bit	0 B	128 B	0 B
+ public metadata	0 B	0 B	0 B
zkSNARK-based	32 B	64 B	$\log_{256}(\ell_{\mathcal{P}})$ + 224 B
+ private bit	0 B	64 B	64 B
+ public metadata	0 B	0 B	0 B

consists of two elements of the underlying pairing-friendly curve's first group and one of the second group, i.e., 192 bytes. Finally, we denote the size of the policy by $\ell_{\mathcal{P}}$ and the byte length of tags by ℓ_{τ}.

We encode the tag within the client's redemption message as an index pointing to the tag's position in the policy. Furthermore, we do not explicitly report the communication of updating or announcing the policy as this strongly depends on the policy update mechanism. To name a few examples, policies can be realized as an interval of integers encoded via their first and last elements, policies can be valid for a long time such that multiple pre-tokens issued to the same party are valid with respect to the same policy, and policies can be self-updating based on pre-defined rules such as "every day, the policy is updated to contain just the current date". In the worst-case, the server sends the whole policy with each issued pre-token, which increases the server's communication complexity in the issuance phase by $\ell_{\tau} \cdot \ell_{\mathcal{P}}$ in all policy-based constructions.

6.3 Comparison

Privacy Pass [17] comes with significantly lower overall communication and computation overhead than our EQS-based and zkSNARK-based constructions (without any metadata) even if we assume that the issuance overhead in our policy-based token schemes vanishes due to the high number of tokens that can be derived from one issuance session. Our MAC-based construction performs comparably to Privacy Pass. If the tokens in Privacy Pass are issued individually, our MAC-based construction performs better communication-wise and only slightly worse computation-wise. If the tokens in Privacy Pass are generated as a batch, i.e., with 32 bytes of server communication per issued token, our MAC-based construction performs only slightly worse communication-wise – the batched token generation only affects the communication but has minimal to no effect on the computation. We note that our EQS-based and zkSNARK-based constructions offer public verifiability, which is not provided by

[2]The size of the pre-token request can straightforwardly be reduced to 48 B as the first element of the pre-token request is hard-coded to be generator G_1 of $\mathbb{G}_1$.

Privacy pass. Furthermore, Privacy Pass puts a high overhead on the issuance server while our work and the work of Faut et al. [21] shift the overhead towards the client and the redemption servers. This makes policy-based tokens particularly interesting for settings, in which a small number of issuance servers is responsible for the generation of tokens that are handled by a large number of clients and verifiers.

When comparing to [21], we observe that our EQS-based and MAC-based constructions provide better computation and communication efficiency in the redemption phase while the construction of Faut et al. [21] outperforms our EQS-based construction in the issuance phase. As the issuance phase's communication amortizes over the more frequently executed token redemptions, we expect the overhead in the issuance phase of our EQS-based construction to be insignificant when compared to its advantage in the redemption phase. While our MAC-based and EQS-based constructions are, therefore, more efficient than the construction of [21], we note that security of our constructions, unlike [21], relies on the random oracle model.

Finally, we compare our constructions among each other. Even though the zkSNARK-based construction outperforms the other two constructions during the issuance phase with regard to both computation and communication and also outperforms the EQS-based construction communication-wise in the redemption phase, it still constitutes a theoretical solution due to the high computation overhead during the redemption phase. The MAC-based construction outperforms the EQS-based construction with regard to both communication and computation and should be chosen if public verifiability is not required. However, the EQS-based construction is still highly efficient – in both phases, each party runs for less than 6 ms in the plain version and below 8 ms if both the private bit and public metadata property is present. Therefore, we assess it to be a viable choice if public verifiability is required. A final interesting insight is that public metadata can be added without any communication overhead.

References

[1] Ghous Amjad, Kevin Yeo, and Moti Yung. 2025. RSA Blind Signatures with Public Metadata. *Proc. Priv. Enhancing Technol.* 2025, 1 (2025), 37–57. doi:10.56553/POPETS-2025-0004

[2] Foteini Baldimtsi, Lucjan Hanzlik, Quan Nguyen, and Aayush Yadav. 2025. Non-interactive Anonymous Tokens with Private Metadata Bit. *IACR Cryptol. ePrint Arch.* (2025), 430. https://eprint.iacr.org/2025/430

[3] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. 2025. On Proving Equivalence Class Signatures Secure from Non-interactive Assumptions. *IACR Cryptol. ePrint Arch.* (2025), 973. https://eprint.iacr.org/2025/973

[4] Fabrice Benhamouda, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2022. Publicly verifiable anonymous tokens with private metadata bit. *IACR Cryptol. ePrint Arch.* (2022), 4. https://eprint.iacr.org/2022/004

[5] Fabrice Benhamouda, Mariana Raykova, and Karn Seth. 2023. Anonymous Counting Tokens. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 14439)*, Jian Guo and Ron Steinfeld (Eds.). Springer, 245–278. doi:10.1007/978-981-99-8724-5_8

[6] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, 31–46. doi:10.1007/3-540-36288-6_3

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3152)*, Matthew K. Franklin (Ed.). Springer, 41–55. doi:10.1007/978-3-540-28628-8_3

[8] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. Compact Multi-signatures for Smaller Blockchains. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11273)*, Thomas Peyrin and Steven D. Galbraith (Eds.). Springer, 435–464. doi:10.1007/978-3-030-03329-3_15

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2248)*, Colin Boyd (Ed.). Springer, 514–532. doi:10.1007/3-540-45682-1_30

[10] Rutchathon Chairattana-Apirom, Nico Döttling, Anna Lysyanskaya, and Stefano Tessaro. 2025. Everlasting Anonymous Rate-Limited Tokens. *IACR Cryptol. ePrint Arch.* (2025), 1030. https://eprint.iacr.org/2025/1030

[11] Melissa Chase, F. Betül Durak, and Serge Vaudenay. 2023. Anonymous Tokens with Stronger Metadata Bit Hiding from Algebraic MACs. In *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 14082)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer, 418–449. doi:10.1007/978-3-031-38545-2_14

[12] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. 2014. Algebraic MACs and Keyed-Verification Anonymous Credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 1205–1216. doi:10.1145/2660267.2660328

[13] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, David Chaum, Ronald L. Rivest, and Alan T. Sherman (Eds.). Plenum Press, New York, 199–203. doi:10.1007/978-1-4757-0602-4_18

[14] David Chaum. 1983. Blind Signature System. In *Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983*, David Chaum (Ed.). Plenum Press, New York, 153.

[15] Arkworks contributors. 2022. `Arkworks` *zkSNARK ecosystem*. https://arkworks.rs

[16] Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. 2023. Snowblind: A Threshold Blind Signature in Pairing-Free Groups. In *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 14081)*, Helena Handschuh and Anna Lysyanskaya (Eds.). Springer, 710–742. doi:10.1007/978-3-031-38557-5_23

[17] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. 2018. Privacy Pass: Bypassing Internet Challenges Anonymously. *Proc. Priv. Enhancing Technol.* 2018, 3 (2018), 164–180. doi:10.1515/POPETS-2018-0026

[18] Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. 2012. Message Authentication, Revisited. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7237)*, David Pointcheval and Thomas Johansson (Eds.). Springer, 355–374. doi:10.1007/978-3-642-29011-4_22

[19] Yevgeniy Dodis and Aleksandr Yampolskiy. 2005. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3386)*, Serge Vaudenay (Ed.). Springer, 416–431. doi:10.1007/978-3-540-30580-4_28

[20] F. Betül Durak, Laurane Marco, Abdullah Talayhan, and Serge Vaudenay. 2024. Non-Transferable Anonymous Tokens by Secret Binding. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 2460–2474. doi:10.1145/3658644.3670338

[21] Dennis Faut, Julia Hesse, Lisa Kohl, and Andy Rupp. 2025. Scalable and Fine-Tuned Privacy Pass from Group Verifiable Random Functions. *IACR Cryptol. ePrint Arch.* (2025), 659. https://eprint.iacr.org/2025/659

[22] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. 2019. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *J. Cryptol.* 32, 2 (2019), 498–546. doi:10.1007/S00145-018-9281-4

[23] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. 2020. Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12106)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, 63–95. doi:10.1007/978-3-030-45724-2_3

[24] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9666)*, Marc

Fischlin and Jean-Sébastien Coron (Eds.). Springer, 305–326. doi:10.1007/978-3-662-49896-5_11

[25] Lucjan Hanzlik. 2023. Non-interactive Blind Signatures for Random Messages. In *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 14008)*, Carmit Hazay and Martijn Stam (Eds.). Springer, 722–752. doi:10.1007/978-3-031-30589-4_25

[26] Ioanna Karantaidou, Omar Renawi, Foteini Baldimtsi, Nikolaos Kamarinakis, Jonathan Katz, and Julian Loss. 2024. Blind Multisignatures for Anonymous Tokens with Decentralized Issuance. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie (Eds.). ACM, 1508–1522. doi:10.1145/3658644.3690364

[27] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous Tokens with Private Metadata Bit. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12170)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, 308–336. doi:10.1007/978-3-030-56784-2_11

[28] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. 1999. Pseudonym Systems. In *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings (Lecture Notes in Computer Science, Vol. 1758)*, Howard M. Heys and Carlisle M. Adams (Eds.). Springer, 184–199. doi:10.1007/3-540-46513-8_14

[29] Simon Masson, Antonio Sanso, and Zhenfei Zhang. 2024. Bandersnatch: a fast elliptic curve built over the BLS12-381 scalar field. *Des. Codes Cryptogr.* 92, 12 (2024), 4131–4143. doi:10.1007/S10623-024-01472-0

[30] Michele Orrù, Stefano Tessaro, Greg Zaverucha, and Chenzhi Zhu. 2024. Oblivious Issuance of Proofs. In *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX (Lecture Notes in Computer Science, Vol. 14928)*, Leonid Reyzin and Douglas Stebila (Eds.). Springer, 254–287. doi:10.1007/978-3-031-68400-5_8

[31] Christian Paquin and Greg Zaverucha. 2023. *U-Prove Cryptographic Specification V1.1 (Revision 5).* Retrieved 2025-04-04 from https://github.com/microsoft/uprove-node-reference/blob/main/doc/U-Prove%20Cryptographic%20Specification%20V1.1%20Revision%205.pdf

[32] Torben P. Pedersen. 1991. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings (Lecture Notes in Computer Science, Vol. 576)*, Joan Feigenbaum (Ed.). Springer, 129–140. doi:10.1007/3-540-46766-1_9

[33] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126. doi:10.1145/359340.359342

[34] Claus-Peter Schnorr. 1989. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings (Lecture Notes in Computer Science, Vol. 435)*, Gilles Brassard (Ed.). Springer, 239–252. doi:10.1007/0-387-34805-0_22

[35] Jacob T. Schwartz. 1980. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM* 27, 4 (1980), 701–717. doi:10.1145/322217.322225

[36] Victor Shoup. 1997. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding (Lecture Notes in Computer Science, Vol. 1233)*, Walter Fumy (Ed.). Springer, 256–266. doi:10.1007/3-540-69053-0_18

[37] Tjerand Silde and Martin Strand. 2022. Anonymous Tokens with Public Metadata and Applications to Private Contact Tracing. In *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 13411)*, Ittay Eyal and Juan A. Garay (Eds.). Springer, 179–199. doi:10.1007/978-3-031-18283-9_9

[38] Richard Zippel. 1979. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings (Lecture Notes in Computer Science, Vol. 72)*, Edward W. Ng (Ed.). Springer, 216–226. doi:10.1007/3-540-09519-5_73

Appendix

A Further Extensions

This section presents further extensions to our constructions providing additional properties.

Non-interactive anonymous tokens. In non-interactive anonymous token protocols [2], tokens are issued to known clients based on their long-time public keys without requiring clients to send a dedicated token request. All three of our constructions, i.e., the one based on equivalence class signatures, the one based on algebraic MACs, and the one based on zkSNARKs, can straight-forwardly be transformed into non-interactive anonymous token schemes. We simply interpret the client requests as long-time public keys of the clients to which tokens are issued. If a client should receive a new pre-token, e.g., when the issuer updates its keys or the metadata associated with a client changes, the issuer creates a new pre-token based on the client's long-term public key. In order to ensure that clients generate different tokens for different issuers or metadata, it becomes necessary to include the metadata and the issuer public key into the computation of the token, i.e., to compute $\delta = \mathsf{sk}_c \cdot \mathcal{H}_2(\mathsf{md}, \mathsf{pk}, \tau)$ instead of $\delta = \mathsf{sk}_c \cdot \mathcal{H}_2(\tau)$.

Non-transferable anonymous tokens. In [20], the authors introduce the notion of non-transferable anonymous tokens. Such token schemes deter clients from transferring tokens to other clients by binding each token to a valuable insurance secret owned by the client that needs to be known for the token redemption. In a nutshell, their protocol realizes the token issuance as a blind signature on the insurance secret and the token redemption as a presentation of the unblinded signature together with a zero-knowledge proof of knowledge of the signed insurance secret, obviously without disclosing the secret. The zero-knowledge proof during the token redemption is interactive, which ensures freshness of the proof and, hence, knowledge of the insurance secret (when assuming that a token seller and token buyer cannot communicate during the token redemption). A similar technique can be applied to all our constructions. We interpret the secret client key as the client's insurance secret and replace the non-interactive zero-knowledge proofs in the token redemption by interactive proofs for the same proof relation.

Tag-hiding token redemption. As every client can use each tag at most once, policy-based anonymous tokens provide weaker unlinkability properties than classical anonymous tokens. For example, consider a verifier observing two redemption sessions. In both sessions, the client utilizes the same tag. As each client can use each tag at most once, the verifier knows that the clients in the two sessions are distinct. Such knowledge cannot be derived in standard anonymous token schemes. Therefore, it could be desirable to extend the notion of policy-based anonymous tokens with a tag hiding-property, in which the tag used for the token redemption remains hidden while still guaranteeing that it is within the specified policy. This property can added to our construction straightforwardly by extending the zero-knowledge proofs in the token redemption to prove that the token has been generated with any of the allowed tags. However, such proofs have a significant impact on the performance. When instantiated with generalized Schnorr proofs, as done in the EQS-based and MAC-based constructions, the proof size grows linearly with the size of the policy. In the zkSNARK-based construction, the proof size stays constant but the client's redemption computation time increases by approximately $(3.2, 12.8, 14.0, 16.8, 18.2)$ % for policies of size $(5, 10, 20, 50, 100)$.

B Proof of Theorem 2

In this section, we prove Theorem 2 stating security of our EQS-based construction (cf. Section 4.3). We do so by showing correctness, freshness, unforgeability, unlinkability and private bit privacy.

B.1 Correctness

By the definition of algorithms Setup, SKeyGen, Req, Issue, Final, and Redeem, the completeness of the zero-knowledge proof systems $(\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify})$, $(\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify})$, $(\mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify})$, and $(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$, and the correctness of the equivalence class signature scheme ES, it follows that if $\mathcal{H}_1(\mathsf{md}) \neq 0, \mathsf{sk}_c \neq 0$ and $\mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}} \neq 0$ then

$$\mathsf{pk}_c^* = \mathsf{sk}_c R^*,\ X^* = \mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}} R^*, M^* = \mathcal{H}_1(\mathsf{md}) \cdot R^*$$
$$\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}^*, \sigma^*) = 1 \text{ for } \mathsf{msg}^* = (R^*, \mathsf{pk}_c^*, M^*, X^*)$$
$$\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify}(\pi_{\mathsf{red}}, (\mathsf{pk}_c^*, \delta, R^*, \mathcal{H}_2(\tau))) = 1$$

for each $(\delta, \tau, \omega = (R^*, \mathsf{pk}_c^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$ produced by the execution of Redeem with the inputs defined as in the correctness definition. As correctness is only defined with respect to $\tau \in \mathcal{P}$ (the initial tag verification cannot fail) and the verification algorithm Verify verifies signatures with respect to message $\mathsf{msg} = (R^*, \mathsf{pk}_c^*, \mathcal{H}_1(\mathsf{md}) \cdot R^*, X^*) = \mathsf{msg}^*$, it follows from the above observation that if $\mathcal{H}_1(\mathsf{md}) \neq 0, \mathsf{sk}_c \neq 0$ and $\mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}} \neq 0$ then

$$\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 1 \text{ and}$$
$$\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) = \mathsf{ok}.$$

As $\mathcal{H}_1$ is a random oracle, sk_c is sampled uniformly random and $\mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}}$ is sampled uniformly random, it holds that $\Pr[\mathcal{H}_1(\mathsf{md}) = 0 \vee \mathsf{sk}_c = 0 \vee \mathsf{sk}_{\mathsf{pb}}^{\mathsf{ok}} = 0] = (1 - (\frac{p-1}{p})^3) \leq \mathsf{negl}(\lambda)$. Hence, we conclude that

$$\Pr[\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0] = (1 - (\frac{p-1}{p})^3) \leq \mathsf{negl} \text{ and}$$
$$\Pr[\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) \neq \mathsf{ok}] = (1 - (\frac{p-1}{p})^3) \leq \mathsf{negl}.$$

Furthermore, ReadBit internally executes Verify. Hence, we conclude that

$$\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 1$$
$$\Leftrightarrow$$
$$\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) \neq \perp.$$

B.2 Freshness

To show freshness, we need to show that the probability, that two tokens created with distinct tags or distinct valid pre-tokens are equal, is negligible. For each of the tokens $\delta_{b,t}$ (with $b, t, \in \{0, 1\}$) considered by the freshness definition, it holds that $\delta_{b,t} = \mathsf{sk}_{c,b} \cdot \mathcal{H}_2(\tau_t)$. Let $r_t = \mathsf{dlog}_{G_1}(\mathcal{H}_2(\tau_t))$. We conclude that

$$\delta_{0,0} = \delta_{1,0} \Leftrightarrow \mathsf{sk}_{c,0} = \mathsf{sk}_{c,1}$$
$$\delta_{0,0} = \delta_{0,1} \Leftrightarrow r_0 = r_1$$
$$\delta_{0,0} = \delta_{1,1} \Leftrightarrow \mathsf{sk}_{c,0} r_0 = \mathsf{sk}_{c,1} r_1$$

By the definition of Req it holds that $\mathsf{sk}_{c,b}$ is sampled uniformly random from $\mathbb{Z}_p$. The tags τ_0 and τ_1 are sampled uniformly random from $\{0, 1\}^{\ell_\tau}$ with the constraint that $\tau_0 \neq \tau_1$. As $\mathcal{H}_2$ is a random oracle, it holds that r_t is distributed uniformly random in $\mathbb{Z}_p$. Furthermore, p is sampled to be exponential in the security parameter. We conclude that

$$\begin{aligned}
&\Pr[\delta_{0,0} = \delta_{0,1} \vee \delta_{0,0} = \delta_{1,0} \vee \delta_{0,0} = \delta_{1,1}] \\
=&\Pr[\mathsf{sk}_{c,0} = \mathsf{sk}_{c,1} \vee r_0 = r_1 \vee \mathsf{sk}_{c,0} r_0 = \mathsf{sk}_{c,1} r_1] \\
=&\Pr[\mathsf{sk}_{c,0} = \mathsf{sk}_{c,1}] + \Pr[r_0 = r_1] \\
&\quad + \Pr[\mathsf{sk}_{c,0} \neq \mathsf{sk}_{c,1} \wedge r_0 \neq r_1 \wedge \mathsf{sk}_{c,0} r_0 = \mathsf{sk}_{c,1} r_1] \\
=&\frac{3}{p} \leq \mathsf{negl}(\lambda).
\end{aligned}$$

B.3 Unforgeability

To show unforgeability, we prove that every PPT adversary $\mathcal{A}$ wins the unforgeability game $\mathsf{Exp}_{\Pi,\mathcal{A},n}^{\mathsf{uf}}$ with probability of at most

$$\Pr[\mathsf{Exp}_{\Pi,\mathcal{A},n}^{\mathsf{uf}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

for Π being the scheme defined in Construction 1, λ being the security parameter and $n > 0$. We conduct the proof via a sequence of hybrid experiments that are defined as follows.

Inline construction. We start by inlining the concrete computations of Π into the security game (where relevant). The game is defined as follows:

```
Experiment: $\mathsf{Exp}_{0,\Pi,\mathcal{A},n}^{\mathsf{uf}}(\lambda)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_{\mathsf{eqs}}, \mathsf{pk}_{\mathsf{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$
(2) $\pi_{\mathsf{pk}} \leftarrow \mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Prove}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{pk}_{\mathsf{eqs}})$
(3) $\forall b \in \{0,1\} : \mathsf{sk}_{\mathsf{pb}}^b \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{pk}_{\mathsf{pb}}^b \leftarrow \mathsf{sk}_{\mathsf{pb}}^b \cdot G_1$
(4) $\mathsf{pk} \leftarrow (\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1),\ \mathsf{sk} \leftarrow (\mathsf{sk}_{\mathsf{eqs}}, \mathsf{sk}_{\mathsf{pb}}^0, \mathsf{sk}_{\mathsf{pb}}^1, \mathsf{pk})$
(5) $(\mathcal{P}, \mathsf{ok}^*, \mathsf{md}^*, \{(\delta_i, \tau_i, \omega_i)\}_{i \in [n]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Is}}, \mathcal{O}_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
(6) $\forall i \in [n] : (R_i, \mathsf{pk}_i, X_i, \sigma_i, \pi_i) \leftarrow \omega_i$
(7) if $\exists i \in [n]$ s.t.
      (a) $\tau \notin \mathcal{P}$ or
      (b) $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, (R_i, \mathsf{pk}_i, \mathcal{H}_1(\mathsf{md}^*) R_i, X_i), \sigma_i) = 0$ or
      (c) $\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify}(\pi_i, (\mathsf{pk}_i, \delta_i, R_i, \mathcal{H}_2(\tau_i)))$ or
      (d) $(\mathsf{ok}^* = 1 \wedge X_i \neq \mathsf{sk}_{\mathsf{pb}}^1 \cdot R_i)$ or
      (e) $(\mathsf{ok}^* = 0 \wedge X_i = \mathsf{sk}_{\mathsf{pb}}^1 \cdot R_i)$
          (Note that our construction assumes $\mathsf{ok}_i$ to be 0 if
          $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}^*, \delta_i, \tau_i, \mathsf{md}^*, \omega_i, \mathsf{pk}) =$ true and $\mathsf{ok}_i \neq 1$.)
    then return 0
(8) if $|\mathcal{P}| \cdot i_{\mathsf{ok}^*,\mathsf{md}^*} \geq n$ or $\exists i \neq j$ s.t. $\delta_i = \delta_j$ return 0  else return 1
...
```

As this hybrid only affects the presentation of the experiment but not the execution, it holds that

$$\Pr[\mathsf{Exp}_{\Pi,\mathcal{A},n}^{\mathsf{uf}}(\lambda) = 1] = \Pr[\mathsf{Exp}_{0,\Pi,\mathcal{A},n}^{\mathsf{uf}}(\lambda) = 1].$$

Panic when observing metadata collisions. In the next hybrid, we change the issuance oracle to keep track of all metadata random oracle responses and abort with output 0 if a collision is detected. We define the corresponding hybrid experiment as follows:

Experiment: $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}(\lambda)$

```
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4),\ \mathcal{M} \leftarrow \emptyset$
(2) ...
$\underline{\mathcal{O}_{\mathsf{Is}}(\mathbf{req}, \mathbf{md}, \mathbf{ok})}$ :
(9) **if** $i_{\mathsf{ok,md}} = \mathbf{null}$, **then** $i_{\mathsf{ok,md}} \leftarrow 1$ **else** $i_{\mathsf{ok,md}} \leftarrow i_{\mathsf{ok,md}} + 1$
(10) **if** $\exists(\mathsf{md}', m') \in \mathcal{M} : \mathsf{md}' \neq \mathsf{md} \land \mathcal{H}_1(\mathsf{md}) = m'$ abort the experiment with
     output 0
(11) $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\mathsf{md}, \mathcal{H}_1(\mathsf{md}))\}$
(12) **return** $\mathsf{resp} \leftarrow \mathsf{Issue}(\mathsf{pp}, \mathsf{req}, \mathsf{md}, \mathsf{ok}, \mathsf{sk})$
...
```

It is clearly evident that the experiments $\mathsf{Exp}^{\mathsf{uf}}_{0,\Pi,\mathcal{A},n}$ and $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}$ only differ, if $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}$ aborts while executing the issuance oracle, i.e., if the issuance oracle detects a random oracle collision. We call this event F. The probability of F is the probability of a hash collision in an ideal hash function, which is well known to be $1 - \frac{p!}{(p-n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}$, where $n_{\mathcal{H}_1}$ is the maximal number of oracle queries to $\mathcal{H}_1$ that can be requested by the adversary.

By applying the difference lemma, we can conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{uf}}_{0,\Pi,\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}(\lambda) = 1]| \leq$$
$$1 - \frac{p!}{(p - n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}.$$

Extract the redemption proofs. In a series of n hybrids, we utilize the extractor $\mathcal{E}_{\mathsf{eq}}$ of proof system $\mathsf{ZKP_{eq}}$ to extract the witnesses of the zero knowledge proofs π_i after the zero-knowledge proofs have been verified correctly. If the extraction fails, we abort the experiment with output 0. For $j \in [2, n+1]$ The j-th hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{uf}}_{j,\Pi,\mathcal{A},n}(\lambda)$

```
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$
(2) $\pi_{\mathsf{pk}} \leftarrow \mathsf{ZKP_{eqs}.Prove}(\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}})$
(3) $\forall b \in \{0,1\} : \mathsf{sk}^b_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{pk}^b_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} \cdot G_1$
(4) $\mathsf{pk} \leftarrow (\mathsf{pk_{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}),\ \mathsf{sk} \leftarrow (\mathsf{sk_{eqs}}, \mathsf{sk}^0_{\mathsf{pb}}, \mathsf{sk}^1_{\mathsf{pb}}, \mathsf{pk})$
(5) $(\mathcal{P}, \mathsf{ok}^*, \mathsf{md}^*, \{(\delta_i, \tau_i, \omega_i)\}_{i\in[n]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Is}}, \mathcal{O}_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
(6) $\forall i \in [n] : (R_i, \mathsf{pk}_i, X_i, \sigma_i, \pi_i) \leftarrow \omega_i$
(7) **if** $\exists i \in [n]$ s.t.
    (a) $\tau \notin \mathcal{P}$ or
    (b) $\mathsf{ES.Verify}(\mathsf{pk_{eqs}}, (R_i, \mathsf{pk}_i, \mathcal{H}_1(\mathsf{md}^*)R_i, X_i), \sigma_i) = 0$ or
    (c) $\mathsf{ZKP_{eq}}(\pi_i, (\mathsf{pk}_i, \delta_i, R_i, \mathcal{H}_2(\tau_i)))$ or
    (d) $(\mathsf{ok}^* = 1 \land X_i \neq \mathsf{sk}^1_{\mathsf{pb}} \cdot R_i)$ or
    (e) $(\mathsf{ok}^* = 0 \land X_i = \mathsf{sk}^1_{\mathsf{pb}} \cdot R_i)$
   **then return** 0
(8) $(\mathsf{sk}_i) \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{eq}}(\pi_i, \mathsf{transcript})$ for $i \in [j]$
(9) **if** $\exists i \in [j] : \mathsf{sk}_i = \perp$ **return** 0
(10) **if** $|\mathcal{P}| \cdot i_{\mathsf{ok}^*,\mathsf{md}^*} \geq n$ **or** $\exists i \neq j$ s.t. $\delta_i = \delta_j$ **return** 0 **else return** 1
...
```

The security games $\mathsf{Exp}^{\mathsf{uf}}_{j,\Pi,\mathcal{A},n}(\lambda)$ and $\mathsf{Exp}^{\mathsf{uf}}_{j-1,\Pi,\mathcal{A},n}(\lambda)$ only differ in the extraction of the j-th zero-knowledge proof and the corresponding success verification. We use the difference lemma to conclude that for $j \in [2, n+1]$

$$|\Pr[\mathsf{Exp}^{\mathsf{uf}}_{j,\Pi,\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{uf}}_{j-1,\Pi,\mathcal{A},n}(\lambda) = 1]|$$
$$\leq \Pr[\mathcal{E}_{\mathsf{eq}} \text{ fails}].$$

The running time of $\mathsf{Exp}^{\mathsf{uf}}_{j,\Pi,\mathcal{A},n}(\lambda)$ increases (roughly) by the running time of $\mathcal{E}_{\mathsf{eq}}$.

We sum up the differences between all of the n freshly introduced hybrids to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}(\lambda) = 1]|$$
$$\leq n \cdot \Pr[\mathcal{E}_{\mathsf{eq}} \text{ fails}].$$

Furthermore, the running time of $\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}(\lambda) = 1$ increases by n times the running time of $\mathcal{E}_{\mathsf{eq}}$ when compared to the running time of $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}(\lambda) = 1$.

Simulate key proofs. In the next hybrid experiment, we utilize the key proof's simulator $\mathcal{S}_{\mathsf{pk}}$ instead of the secret key $\mathsf{sk_{eqs}}$ to compute the key proof π_{pk}. The simulator receives the proof statement $(\mathsf{pp}, \mathsf{pk_{eqs}})$, has the capability to program the random oracle and creates a valid zero-knowledge proof π_{pk} except with negligible probability. The hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$

```
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$
(2) $\pi_{\mathsf{pk}} \leftarrow \mathcal{S}_{\mathsf{pk}}(\mathsf{pp}, \mathsf{pk_{eqs}})$
(3) ...
```

In $\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}$ the key proof is always valid due to the completeness property of the proof system. Hence, $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ and $\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}$ only differ if $\mathcal{S}_{\mathsf{pk}}$ in $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ is not successful. We use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}(\lambda) = 1]|$$
$$\leq \Pr[\mathcal{S}_{\mathsf{pk}} \text{ fails}].$$

The runtime of $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ when compared to $\mathsf{Exp}^{\mathsf{uf}}_{n+1,\Pi,\mathcal{A},n}$ increases by the runtime of $\mathcal{S}_{\mathsf{pk}}$.

Reduction to signature unforgeability. In the final security experiment, $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$, we bind the adversary's success probability via a reduction to the unforgeability property of the equivalence class signature scheme. More precisely, we build an adversary $\mathcal{B}$ attacking the unforgeability (EUF-CMA) property of the equivalence class signature scheme ES by internally simulating an execution of $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$. Adversary $\mathcal{B}$ is defined as follows.

$\mathcal{B}$ is executed with input $(\mathsf{pp}, \mathsf{pk}')$ for $\mathsf{pp} \leftarrow \mathsf{BGGen}(1^\lambda)$ and $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{ES.KeyGen}(1^4)$, has oracle access to $\mathcal{O}_{\mathsf{Sig}}$ and simulates an execution of $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ with adversary $\mathcal{A}$.

- Upon receiving $(\mathsf{pp}, \mathsf{pk}')$, $\mathcal{B}$ samples $\mathsf{sk}^b_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p$ and computes $\mathsf{pk}^b_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} \cdot G_1$ for $b \in \{0,1\}$.
- $\mathcal{B}$ computes $\pi_{\mathsf{pk}} \leftarrow \mathcal{S}_{\mathsf{pk}}(\mathsf{pp}, \mathsf{pk}')$.
- $\mathcal{B}$ defines $\mathsf{pk_{eqs}} \leftarrow \mathsf{pk}'$ and $\mathsf{pk} \leftarrow (\mathsf{pk_{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$.
- $\mathcal{B}$ executes $(\mathcal{P}, \mathsf{ok}^*, \mathsf{md}^*, \{(\delta_i, \tau_i, (R_i, \mathsf{pk}_i, X_i, \sigma_i, \pi_i))\}_{i\in[n]}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Is}}, \mathcal{O}_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
  - It answers oracle queries to $\mathcal{O}_{\mathsf{RB}}$ as defined in experiment $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$.
  - It answers oracle queries to $\mathcal{O}_{\mathsf{Is}}$ as follows:

```
(11) Parse $(\mathsf{pk}_c, \pi_{\mathsf{req}}) \leftarrow \mathsf{req}$
(12) **if** $i_{\mathsf{ok,md}} = \mathbf{null}$   $i_{\mathsf{ok,md}} \leftarrow 1$, $\mathcal{K}_{\mathsf{ok,md}} \leftarrow \{\mathsf{pk}_c\}$
     **else** $i_{\mathsf{ok,md}} \leftarrow i_{\mathsf{ok,md}} + 1$, $\mathcal{K}_{\mathsf{ok,md}} \leftarrow \mathcal{K}_{\mathsf{ok,md}} \cup \{\mathsf{pk}_c\}$
(13) **if** $\exists(\mathsf{md}', m') \in \mathcal{M} : \mathsf{md}' \neq \mathsf{md} \land \mathcal{H}_1(\mathsf{md}) = m'$ abort the
     experiment with output 0
(14) $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\mathsf{md}, \mathcal{H}_1(\mathsf{md}))\}$
(15) If $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ then output $\perp$.
(16) Sample $v \xleftarrow{\$} \mathbb{Z}^*_p$ and define $R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c$ and $X \leftarrow$
     $v \cdot \mathsf{pk}^{\mathsf{ok}}_{\mathsf{pb}}$.
(17) Compute $\pi_{\mathsf{is}} \leftarrow \mathsf{ZKP_{eq\text{-}or}.Prove}((v, \mathsf{ok}), (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}))$.
```

```
    (18) Compute $m \leftarrow \mathcal{H}_1(\mathsf{md})$ and define $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$.
    (19) Compute $\sigma \leftarrow O_{\mathsf{Sig}}(\mathsf{msg})$.
    (20) Output $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
• $\mathcal{B}$ continues the execution of $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ which yields output $o$.
• If $o = 0$, $\mathcal{B}$ returns $\bot$.
• If $\mathsf{ok}^* = 0$ and $\exists i \in [n]$ such that $X_i \neq \mathsf{sk}^0_{\mathsf{pb}} R_i$ then $\mathcal{B}$ returns
  $((R_i, \mathsf{pk}_i, \mathcal{H}(\mathsf{md}^*)R_i, X_i), \sigma_i)$.
  (As $o = 0$ and $\mathsf{ok}^* = 0$, it holds that $X_i \neq \mathsf{sk}^1_{\mathsf{pb}} R_i$.)
• $\mathcal{B}$ defines $\mathcal{K}^* \leftarrow \{j \mid j \in [n] \wedge \mathsf{sk}_j \cdot G_1 \notin \mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}\}$.
• $\mathcal{B}$ picks the first element $j^*$ in $\mathcal{K}^*$ and returns
  $((R_{j^*}, \mathsf{pk}_{j^*}, \mathcal{H}_1(\mathsf{md}^*)R_{j^*}, X_{j^*}), \sigma_i)$.
```

First, we note that adversary $\mathcal{B}$ perfectly simulates the experiment $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$ to $\mathcal{A}$. This is due to the fact that the public parameters, the public key and oracle responses are computed exactly as in $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$ even though the signatures are not computed by $\mathcal{B}$ itself. Furthermore, the computation of $\mathcal{K}_{\cdot,\cdot}$ is oblivious to $\mathcal{A}$.

Next, we look at the different outputs of $\mathcal{B}$. If $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$ would return 0 ($o = 0$), $\mathcal{B}$ returns $\bot$, which cannot be a valid forgery.

If $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$ would return 1 ($o = 1$), $\mathsf{ok}^* = 0$, and there exists $i \in [n]$ such that $X_i \neq \mathsf{sk}^0_{\mathsf{pb}} R_i$ then $\mathcal{B}$ returns $(\mathsf{msg}_i = (R_i, \mathsf{pk}_i, \mathcal{H}_1(\mathsf{md}^*)R_i, X_i), \sigma_i)$. We know that $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}_i, \sigma_i) = 1$ and $X_i \neq \mathsf{sk}^1_{\mathsf{pb}} R_i$ as this is a necessary condition for $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)$ outputting 1. From the former, we can conclude that σ_i is a valid signature on msg_i. The latter ensures that $X_i \neq \mathsf{sk}^b_{\mathsf{pb}} R_i$ for $b \in \{0, 1\}$. As $\mathcal{B}$ never queries O_{Sig} with any message $\mathsf{msg}' = (R', \cdot, \cdot, X')$ such that $X' \neq \mathsf{sk}^b_{\mathsf{pb}}$ for $b \in \{0, 1\}$ ($R' = rG_1, X' = r \cdot \mathsf{pk}^{\mathsf{ok}}_{\mathsf{pb}} = (r \cdot \mathsf{sk}^{\mathsf{ok}}_{\mathsf{pb}})G_1$ $= \mathsf{sk}^{\mathsf{ok}}_{\mathsf{pb}} \cdot R'$), it follows that $(\mathsf{msg}_i, \sigma_i)$ is a valid forgery.

To analyze the remaining case, we first observe that if $o = 1$, there are at least $n' \geq \lceil \frac{n}{|\mathcal{P}|} \rceil$ distinct client secret keys sk_i. This is due to the fact that all provided tokens are distinct, that $\tau_i \in \mathcal{P}$ for each $i \in [n]$, and that $\delta_i = \mathsf{sk}_i \cdot \mathcal{H}_2(\tau_i)$ for each $i \in [n]$. Note that collisions of the form $\mathcal{H}_2(\tau_i) = \mathcal{H}_2(\tau_j)$ or $\mathsf{sk}_i \cdot \mathcal{H}_2(\tau_i) = \mathsf{sk}_j \cdot \mathcal{H}_2(\tau_j)$ for $i \neq j$, do not affect this observation as they reduce the number of tokens that can be provided per client secret key.

Next, we observe that if $o = 1$ there has to be at least one element in $\mathcal{K}^*$. As $n > i_{\mathsf{ok}^*,\mathsf{md}^*} \cdot |\mathcal{P}|$, and $n' \geq \lceil \frac{n}{|\mathcal{P}|} \rceil$ it follows that $n' > i_{\mathsf{ok}^*,\mathsf{md}^*}$. Furthermore, it holds that $\mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}$ has size $i_{\mathsf{ok}^*,\mathsf{md}^*}$ and, hence, that there exists at least one sk_j such that $\mathsf{sk}_j \cdot G_1 \notin \mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}$.

Let $m^* = \mathcal{H}_1(\mathsf{md}^*)$. What is left is to show that for each $j^* \in \mathcal{K}^*$ the message $\mathsf{msg}^* = (R_{j^*}, \mathsf{pk}_{j^*}, m^* R_{j^*}, X_{j^*})$ and signature σ_{j^*} constitute a valid forgery. First, we observe that σ_{j^*} is a valid signature on msg^*. This is due to the fact that $j^* \in [n]$ and that $\mathsf{ES.Verify}(\mathsf{pk}_s, (R_i, \mathsf{pk}_i, m^* R_i, X_i), \sigma_i) = 1$ for each $i \in [n]$ is a necessary condition for $o = 1$. Second, we show that the message class of msg^* has not been queried before. To see this note that for every queried $\mathsf{msg} = (rG_1, r \cdot \mathsf{pk}'_c, (m^* \cdot r)G_1, (\mathsf{sk}^{\mathsf{ok}^*}_{\mathsf{pb}} \cdot r)G_1)$ it holds that $(\mathsf{pk}' = \mathsf{sk}' G_1) \in \mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}$ (sk' is not known). This is due to the way $\mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}$ is defined and the fact that the experiment aborts if any collision of md^* is detected. As we pick j^* such that $\mathsf{sk}_{j^*} \cdot G_1 \notin \mathcal{K}_{\mathsf{ok}^*,\mathsf{md}^*}$ it needs to hold that no message $(rG_1, r\mathsf{pk}_i, m^*(rG_1), \mathsf{sk}^{\mathsf{ok}^*}_{\mathsf{pb}}(rG_1))$ with $\mathsf{pk}_i = \mathsf{sk}_{j^*} G_1$ has been queried before.

Based on the above observations, we conclude that $\mathcal{B}$ outputs a valid forgery and, hence, wins the signature unforgeability experiment with the probability as $\Pr[\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda)] = 1$. Therefore, we conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}(\lambda) = 1] \leq \mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$$

where $\mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$ is the maximal advantage of any adversary, running in time roughly equivalent to $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$, in breaking the security of the EQS-scheme.

Putting all steps together. By combining all hybrids, we can conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A},n}(\lambda) = 0] \leq (1 - \frac{p!}{(p - n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}) + n \cdot \Pr[\mathcal{E}_{\mathsf{eq}} \text{ fails}]$$
$$+ \Pr[\mathcal{S}_{\mathsf{pk}} \text{ fails}]) + \mathsf{Adv}_{\mathsf{eqs\text{-}uf}} \leq \mathsf{negl}(\lambda)$$

where $n_{\mathcal{H}_1}$ is the number of random oracle queries to $\mathcal{H}_1$ available to the adversary and $\mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$ is with respect to an adversary running in time approximately equal to $\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A},n}(\lambda)$ plus n times the running time of $\mathcal{E}_{\mathsf{eq}}$ and once the running time of $\mathcal{S}_{\mathsf{pk}}$.

B.4 Unlinkability

To show unlinkability, in particular 2-unlinkability, we prove that for every PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}(\lambda) = 1] \leq \frac{2}{n} + \mathsf{negl}(\lambda).$$

We construct our proof based on a series of hybrid experiments. Let $\mathsf{Exp}^{\mathsf{ul}}_0 = \mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}$. The subsequent hybrids are defined as follows:

Inline construction. In the first game, we inline our construction into the security game. The resulting game is defined as follows.

Experiment: $\mathsf{Exp}^{\mathsf{ul}}_1(\lambda)$

```
(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $Q_{\mathsf{redeem}} \leftarrow \emptyset, j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  if $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}}) = 0$ return 0
(4)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(5)  if $|Q| < n \vee \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ return 0
(6)  if $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \bot\})$ return 0
(7)  if $Q$ contains duplicates return 0
(8)  $\forall i \in Q$ :
(9)      $(\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(10)     $r \xleftarrow{\$} \mathbb{Z}^*_p$, $\mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}$, $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(11)     $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*$, $\sigma^* \leftarrow \mathsf{ES.ChgRp}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma, r)$
(12)     $\pi_{\mathsf{red}} \leftarrow \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}((\mathsf{sk}), (\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau)))$
(13)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(14) $k \xleftarrow{\$} Q$, pick a random permutation $\phi$ of $Q$
(15) $k^* \leftarrow \mathcal{A}(2, s_1, \phi(k), \{\delta_{\phi(i)}, \omega_{\phi(i)}\}_{i \in Q})$
(16) if $k = k^*$ return 1 else return 0

$O_{\mathsf{Req}}(j, \mathsf{md})$ :
(16) $j \leftarrow j + 1$
(17) $\mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p, \mathsf{pk}_j \leftarrow \mathsf{sk}_j G_1$
(18) $\pi_{\mathsf{req}} \leftarrow \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}(\mathsf{sk}_j, (G_1, \mathsf{pk}_j))$
(19) return $(\mathsf{pk}_j, \pi_{\mathsf{req}})$

$O_{\mathsf{Is}}(i, \mathsf{resp})$ :
(20) if $\mathsf{pk}_i = \mathbf{null} \vee \rho_i \neq \mathbf{null}$ return $\bot$
(21) $(R, \mathsf{pk}'_c, X, \sigma, \pi_{\mathsf{is}}) \leftarrow \mathsf{resp}, \mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, \mathcal{H}_1(\mathsf{md}_i)R, X)$
(22) if $\mathsf{pk}'_c \neq \mathsf{sk}_i R \vee \mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma) = 0$
         $\vee\ \mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify}(\pi_{\mathsf{is}}, (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})) = 0$
             $\rho_i \leftarrow \bot$, return 0
(23) $\rho_i \leftarrow (\mathsf{msg}, \mathsf{sk}_i, \sigma)$, return 1

$O_{\mathsf{Rd}}(i, \tau)$ :
```

```
(24) if $\rho_i \in \{\mathbf{null}, \bot\} \vee (i,\tau) \in Q_{\mathsf{redeem}}$ return $\bot$
(25) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i,\tau)\}, (\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(26) $r \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}, \delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(27) $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*, \sigma^* \leftarrow \mathsf{ES.ChgRp}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma, r)$
(28) $\pi_{\mathsf{red}} \leftarrow \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}((\mathsf{sk}), (\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau)))$
(29) return $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

This step does not affect the output distribution. It follows that

$$\Pr[\mathsf{Exp}_1^{\mathsf{ul}}(\lambda) = 1] = \Pr[\mathsf{Exp}_0^{\mathsf{ul}}(\lambda) = 1]$$

Simulate request proofs. In a first series of hybrids, we utilize the discrete logarithm proof's simulator $\mathcal{S}_{\mathsf{dlog}}$ to simulate the proofs created by the request oracle, one after another. The result of the hybrid series is defined as follows:

```
Experiment: $\mathsf{Exp}_2^{\mathsf{ul}}(\lambda)$
...
$O_{\mathsf{Req}}(j, \mathbf{md})$ :
(16) $j \leftarrow j + 1$
(17) $\mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p, \mathsf{pk}_j \leftarrow \mathsf{sk}_j G_1$
(18) $\pi_{\mathsf{req}} \leftarrow \mathcal{S}_{\mathsf{dlog}}(G_1, \mathsf{pk}_j)$
(19) return $(\mathsf{pk}_j, \pi_{\mathsf{req}})$
...
```

Note that the request proofs in the previous hybrid ($\mathsf{Exp}_1^{\mathsf{ul}}$) are always valid as the client public key is generated honestly. Therefore, it becomes evident that the output distributions of two consecutive experiments only differ if the newly included proof simulation is not successful. Therefore, we can use that difference lemma to conclude that

$$|\Pr[\mathsf{Exp}_2^{\mathsf{ul}}(\lambda) = 1] - \Pr[\mathsf{Exp}_1^{\mathsf{ul}}(\lambda) = 1]| \\ \leq n_{\mathsf{req}} \cdot \Pr[\mathcal{S}_{\mathsf{dlog}} \text{ fails}]$$

where n_{req} is the number of allowed request oracle queries. The runtime of the experiment increases approximately by n_{req} times the runtime of $\mathcal{S}_{\mathsf{dlog}}$.

Extract issuance proofs. In a series of hybrids, we utilize the extractor $\mathcal{E}_{\mathsf{eq\text{-}or}}$ of proof system $\mathsf{ZKP}_{\mathsf{eq\text{-}or}}$ to extract the witnesses of the zero knowledge proofs π_{is} one after another. The extractor receives the randomness of the experiment execution μ (including the adversary's random tape), has access to $\mathcal{A}$, and extracts the witnesses (v, ok). The witnesses either satisfy the proof relation of $\mathsf{ZKP}_{\mathsf{eq\text{-}or}}$ or are all $\bot$. In the latter case, we abort the experiment with output 0. The result of the hybrid series is defined as follows:

```
Experiment: $\mathsf{Exp}_3^{\mathsf{ul}}(\lambda; \mu)$
...
$O_{\mathsf{Is}}(i, \mathbf{resp})$ :
(20) if $\mathsf{pk}_i = \mathbf{null} \vee \rho_i \neq \mathbf{null}$ return $\bot$
(21) $(R, \mathsf{pk}_c', X, \sigma, \pi_{\mathsf{is}}) \leftarrow \mathsf{resp}, \mathsf{msg} \leftarrow (R, \mathsf{pk}_c', \mathcal{H}_1(\mathsf{md}_i)R, X)$
(22) if $\mathsf{pk}_c' \neq \mathsf{sk}_i R \ \vee\ \mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma) = 0$
         $\vee\ \mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify}(\pi_{\mathsf{is}}, (R, X, G_1, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1)) = 0$
         $\rho_i \leftarrow \bot$, return 0
(23) $(v, \mathsf{ok}_i) \leftarrow \mathcal{E}_{\mathsf{eq\text{-}or}}^{\mathcal{A}}(\mu)$, if $(r, \mathsf{ok}_i) = \bot^2$ abort experiment with 0
(24) $\rho_i \leftarrow (\mathsf{msg}, \mathsf{sk}_i, \sigma)$, return 1
...
```

It is clear that the output distributions of two consecutive experiments only differ if the newly added extraction is not successful. It follows from the difference lemma that

$$|\Pr[\mathsf{Exp}_3^{\mathsf{ul}}(\lambda) = 1] - \Pr[\mathsf{Exp}_2^{\mathsf{ul}}(\lambda) = 1]| \\ \leq n_{\mathsf{is}} \cdot \Pr[\mathcal{E}_{\mathsf{eq\text{-}or}} \text{ fails}]$$

where n_{is} is the number of allowed issuance oracle queries. The runtime of the experiment increases approximately by n_{is} times the runtime of $\mathcal{E}_{\mathsf{eq\text{-}or}}$.

Extract equivalence class signature key proof. In the next hybrid, we utilize extractor $\mathcal{E}_{\mathsf{eqs}}$ of the equivalence class signature key proof system $\mathsf{ZKP}_{\mathsf{eqs}}$ to extract the secret key $\mathsf{sk}_{\mathsf{eqs}}$ corresponding to $\mathsf{pk}_{\mathsf{eqs}}$. The extractor either outputs a witness $\mathsf{sk}_{\mathsf{eqs}}$ that satisfies the proof relation or $\bot$. In the latter case, we abort the experiment with output 0.

```
Experiment: $\mathsf{Exp}_4^{\mathsf{ul}}(\lambda; \mu)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ Q_{\mathsf{redeem}} \leftarrow \emptyset,\ j \leftarrow 0$
(2) $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3) if $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ return 0
(4) $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}_{\mathsf{eqs}}^{\mathcal{A}}(\mu)$, if $\mathsf{sk}_{\mathsf{eqs}} = \bot$ return 0
(5) $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6) if $|Q| < n \ \vee\ \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ return 0
(7) if $\exists i \in Q$ s.t. $((i,\tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \bot\})$ return 0
(8) if $Q$ contains duplicates return 0
(9) ...
...
```

The output distribution of the new experiment differs only from the previous one if the newly added extraction is not successful. We can apply the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}_4^{\mathsf{ul}}(\lambda) = 1] - \Pr[\mathsf{Exp}_3^{\mathsf{ul}}(\lambda) = 1]| \leq \Pr[\mathcal{E}_{\mathsf{eqs}} \text{ fails}].$$

The runtime of the experiment increases by the runtime of $\mathcal{E}_{\mathsf{eqs}}$.

Simulate redemption proofs. In another series of hybrids, we replace all redemption proofs (within the main thread and within the redemption oracle) by simulations crafted by $\mathcal{S}_{\mathsf{eq}}$, the simulator of the redemption proof system $\mathsf{ZKP}_{\mathsf{eq}}$. The final hybrid of the series is defined as follows:

```
Experiment: $\mathsf{Exp}_5^{\mathsf{ul}}(\lambda; \mu)$
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ Q_{\mathsf{redeem}} \leftarrow \emptyset,\ j \leftarrow 0$
(2) $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3) if $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}}) = 0$ return 0
(4) $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}_{\mathsf{eqs}}^{\mathcal{A}}(\mu)$, if $\mathsf{sk}_{\mathsf{eqs}} = \bot$ return 0
(5) $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6) if $|Q| < n \ \vee\ \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ return 0
(7) if $\exists i \in Q$ s.t. $((i,\tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \bot\})$ return 0
(8) if $Q$ contains duplicates return 0
(9) $\forall i \in Q$ :
(10)     $(\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(11)     $r \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}, \delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(12)     $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*, \sigma^* \leftarrow \mathsf{ES.ChgRp}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma, r)$
(13)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(14)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(15) ...
...
$O_{\mathsf{Rd}}(i, \tau)$ :
(26) if $\rho_i \in \{\mathbf{null}, \bot\} \vee (i,\tau) \in Q_{\mathsf{redeem}}$ return $\bot$
(27) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i,\tau)\}, (\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(28) $r \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}, \delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(29) $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*, \sigma^* \leftarrow \mathsf{ES.ChgRp}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma, r)$
(30) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(31) return $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

Note that the redemption proofs in $\mathsf{Exp}_4^{\mathsf{ul}}$ are always valid as the tokens are created honestly. Therefore, it becomes evident that the output distributions of two consecutive experiments only differ if the newly included proof simulation is not successful. Therefore, we can use that difference lemma to conclude that

$$|\Pr[\mathsf{Exp}_5^{\mathsf{ul}}(\lambda) = 1] - \Pr[\mathsf{Exp}_4^{\mathsf{ul}}(\lambda) = 1]| \\ \leq (n_{\mathsf{red}} + n) \cdot \Pr[\mathcal{S}_{\mathsf{eq}} \text{ fails}]$$

where n_{red} is the number of allowed redemption oracle queries. The runtime of the experiment increases approximately by $(n_{\mathsf{red}} + n)$ times the runtime of $\mathcal{S}_{\mathsf{eq}}$.

Re-sign tokens. We now sign the re-randomized messages directly instead of deriving the signature from the pre-token's signature.

```
Experiment: $\mathsf{Exp}_6^{\mathsf{ul}}(\lambda; \mu)$

(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ Q_{\mathsf{redeem}} \leftarrow \emptyset,\ j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** 0
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}_{\mathsf{eqs}}^{\mathcal{A}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** 0
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  **if** $|Q| < n \vee \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ **return** 0
(7)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** 0
(8)  **if** $Q$ contains duplicates **return** 0
(9)  $\forall i \in Q$ :
(10)     $(\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(11)     $r \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}, \delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(12)     $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*, \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(13)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(14)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(15) ...
...
$\underline{O_{\mathbf{Rd}}(i, \tau)}$ :
(26) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(27) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, (\mathsf{msg}, \mathsf{sk}, \sigma) \leftarrow \rho_i$
(28) $r \xleftarrow{\$} \mathbb{Z}_p^*, \mathsf{msg}^* \leftarrow r \cdot \mathsf{msg}, \delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(29) $(R^*, \mathsf{pk}^*, \cdot, X^*) \leftarrow \mathsf{msg}^*, \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(30) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(31) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

From the perfect signature adaption property of the equivalence class signature scheme EQS, it follows that

$$\Pr[\mathsf{Exp}_6^{\mathsf{ul}}(\lambda) = 1] = \Pr[\mathsf{Exp}_5^{\mathsf{ul}}(\lambda) = 1].$$

The runtime of the experiment stays approximately the same.

Re-compute messages. Next, we re-compute the messages msg^* from freshly sampled randomness instead of re-randomizing the existing message msg. The re-randomized client public key pk^* is computed based on the initial client public key pk_j instead of the client secret key sk. We make sure to maintain the relations between the message components, i.e., the discrete logarithm of the components two, three and four with respect to the first component.

```
Experiment: $\mathsf{Exp}_7^{\mathsf{ul}}(\lambda; \mu)$

(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ Q_{\mathsf{redeem}} \leftarrow \emptyset,\ j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** 0
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}_{\mathsf{eqs}}^{\mathcal{A}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** 0
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  **if** $|Q| < n \vee \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ **return** 0
(7)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** 0
(8)  **if** $Q$ contains duplicates **return** 0
(9)  $\forall i \in Q$ :
(10)     $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(11)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
(12)     $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(13)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(14)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(15) ...
...
$\underline{O_{\mathbf{Rd}}(i, \tau)}$ :
(26) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(27) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, (\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i$
(28) $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(29) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
(30) $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(31) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(32) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

As $r \cdot v$ for $r, v \xleftarrow{\$} \mathbb{Z}_p^*$ is equally distributed to $r^* \xleftarrow{\$} \mathbb{Z}_p^*$, it holds that R^* is equally distributed in both $\mathsf{Exp}_7^{\mathsf{ul}}$ and $\mathsf{Exp}_6^{\mathsf{ul}}$. Furthermore, the other message components are distributed identically as well, i.e., the discrete logarithm with respect to the first message is the same in both $\mathsf{Exp}_7^{\mathsf{ul}}$ and $\mathsf{Exp}_6^{\mathsf{ul}}$. Therefore, it follows that the re-computed messages in $\mathsf{Exp}_7^{\mathsf{ul}}$ are distributed equally to the re-randomized messages in $\mathsf{Exp}_6^{\mathsf{ul}}$. We conclude that

$$\Pr[\mathsf{Exp}_7^{\mathsf{ul}}(\lambda) = 1] = \Pr[\mathsf{Exp}_6^{\mathsf{ul}}(\lambda) = 1].$$

Panic at oracle collisions. In the next experiment, we check if the tag hash computed by $\mathcal{H}_2$ in the main thread collides with a tag hash computed within the redemption oracle (for a different tag) and abort the experiment with output 0 if this is the case.

```
Experiment: $\mathsf{Exp}_8^{\mathsf{ul}}(\lambda; \mu)$

(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (Q_{\mathsf{redeem}}, C) \leftarrow \emptyset^2,\ j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{pb}}^0, \mathsf{pk}_{\mathsf{pb}}^1), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** 0
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}_{\mathsf{eqs}}^{\mathcal{A}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** 0
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  **if** $|Q| < n \vee \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ **return** 0
(7)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** 0
(8)  **if** $Q$ contains duplicates **return** 0
(9)  **if** $\exists i \in Q$ s.t. $(i, \cdot, \mathcal{H}_2(\tau)) \in C$ **return** 0
(10) $\forall i \in Q$ :
(11) ...
...
$\underline{O_{\mathbf{Rd}}(i, \tau)}$ :
(27) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(28) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
(29) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(30) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
(31) $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(32) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(33) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

It is obvious that the executions of $\mathsf{Exp}_8^{\mathsf{ul}}$ and $\mathsf{Exp}_7^{\mathsf{ul}}$ only differ if $\exists i \in Q$ s.t. $(i, \cdot, \mathcal{H}_2(\tau)) \in C$ and $\nexists i \in Q$ s.t. $(i, \tau) \in Q_{\mathsf{redeem}}$. We call this event F. To analyze the probability of event F, we first observe that for every element $(i, \tau', \mathcal{H}_2(\tau')) \in C$ there exists an element $(i, \tau') \in Q_{\mathsf{redeem}}$ (and vice versa). Therefore, we can conclude that event F can only happen if there exists a tuple $(i, \tau', \mathcal{H}_2(\tau)) \in C$ such that $\tau' \neq \tau$. In other words, F can only happen, if the adversary finds a collision of $\mathcal{H}_2$. As $\mathcal{H}_2$ samples its output uniformly random, the probability of F is the probability of a hash collision in an ideal hash function, which is well known to be $1 - \frac{p!}{(p-n_{\mathcal{H}_2})! \cdot p^{n_{\mathcal{H}_2}}}$, where $n_{\mathcal{H}_2}$ is the maximal number of oracle queries to $\mathcal{H}_2$ that can be requested by the adversary[3]. We use the difference lemma to conclude that

$$\begin{aligned}|\Pr[\mathsf{Exp}_8^{\mathsf{ul}}(\lambda) = 1] - \Pr[\mathsf{Exp}_7^{\mathsf{ul}}(\lambda) = 1]|\\ \leq 1 - \frac{p!}{(p - n_{\mathcal{H}_2})! \cdot p^{n_{\mathcal{H}_2}}}.\end{aligned}$$

Decouple public keys from tokens. In a series of n_{req} hybrids (n_{req} is the number of allowed queries to the request oracle), we remove the correlation between the client public keys and tokens by randomly sampling the public keys instead of using the secret key. For $t' \in [0, n_{\mathsf{req}}]$, we define the series of hybrids as below. Note that $\mathsf{Exp}_{9,0}^{\mathsf{ul}} = \mathsf{Exp}_8^{\mathsf{ul}}$.

```
Experiment: $\mathsf{Exp}_{9,t'}^{\mathsf{ul}}(\lambda; \mu)$
...
$\mathcal{O}_{\mathsf{Req}}(j, \mathsf{md})$ :
(19) $j \leftarrow j + 1$
(20) $\mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p$
(21) if $j \leq t'$ $\mathsf{pk}_j \xleftarrow{\$} \mathbb{G}_1$ else $\mathsf{pk}_j \leftarrow \mathsf{sk}_j G_1$
(22) $\pi_{\mathsf{req}} \leftarrow \mathcal{S}_{\mathsf{dlog}}(G_1, \mathsf{pk}_j)$, return $(\mathsf{pk}_j, \pi_{\mathsf{req}})$
...
```

We show indistinguishability between $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$ and $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$ (for $t \in [n_{\mathsf{req}}]$) by reducing to the DDH assumption in $\mathbb{G}_1$. More precisely, we build a DDH distinguisher $\mathcal{B}$ that receives a bilinear DDH challenge $\mathsf{ddh} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2, A = aG_1, B = bG_1, C = cG_1)$ with $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2) \leftarrow \mathsf{BGGen}(1^\lambda)$, internally executes an experiment with $\mathcal{A}$, and outputs a bit guess distinguishing real DDH tuples ($c = ab$) from random ones ($c \xleftarrow{\$} \mathbb{Z}_p$). Formally, we define $\mathcal{B}$ as follows:

```
• $\mathcal{B}$ receives $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2, A, B, C)$ and defines $\mathsf{pp}' \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2)$.
• If $B$ is not a generator of $\mathbb{G}_1$, i.e., if $B = 0$, $\mathcal{B}$ aborts and outputs 0.
• $\mathcal{B}$ defines $\Gamma \leftarrow \emptyset$.
• $\mathcal{B}$ executes $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}(\lambda; \mu)$ with randomly sampled random tape $\mu$ and the following modifications.
    – When receiving a query $(\tau)$ for $\mathcal{H}_2$ (from $\mathcal{A}$ or from itself), $\mathcal{B}$ checks if there is an element $(\tau, \beta_\tau)$ in $\mathcal{B}$. If this is the case, $\mathcal{B}$ responds with $\beta_\tau B$. Otherwise, $\mathcal{B}$ samples $\beta_\tau \xleftarrow{\$} \mathbb{Z}_p$, computes $D = \beta_\tau B$, stores $(\tau, \beta_\tau)$ and responds with $D$.
    – $\mathcal{B}$ adapts the experiment's main thread as follows:
      (1) $\mathsf{pp} \leftarrow \mathsf{pp}', \mathcal{Q}_{\mathsf{redeem}}, C \leftarrow \emptyset^2, j \leftarrow 0$
      (2) ...
      (9) $\forall i \in Q$ :
      (10)     $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
      (11)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
      (12)     if $i = t$ compute $(\cdot) \leftarrow \mathcal{H}_2(\tau)$, pick $(\tau', \beta_{\tau'})$ from $\Gamma$
               such that $\tau = \tau'$ and compute $\delta \leftarrow \beta_{\tau'} C$
               else $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
               $\sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
      (13)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
      (14)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
      (15) ...
    – $\mathcal{B}$ answers request oracle queries $(j, \mathsf{md})$ as follows:
      (18) $j \leftarrow j + 1$
      (19) $\mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p$
      (20) if $j < t$ $\mathsf{pk}_j \xleftarrow{\$} \mathbb{G}_1$
           elseif $j = t$ $\mathsf{pk}_j \leftarrow A$
           else $\mathsf{pk}_j \leftarrow \mathsf{sk}_j G_1$
      (21) $\pi_{\mathsf{req}} \leftarrow \mathcal{S}_{\mathsf{dlog}}(G_1, \mathsf{pk}_j)$, return $(\mathsf{pk}_j, \pi_{\mathsf{req}})$
    – $\mathcal{B}$ answers issuance oracle queries as defined in $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$
    – $\mathcal{B}$ answers redemption oracle queries $(i, \tau)$ as follows:
      (27) if $\rho_i \in \{\mathbf{null}, \bot\} \vee (i, \tau) \in \mathcal{Q}_{\mathsf{redeem}}$ return $\bot$
      (28) $\mathcal{Q}_{\mathsf{redeem}} \leftarrow \mathcal{Q}_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
      (29) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
      (30) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
      (31) if $i = t$ compute $(\cdot) \leftarrow \mathcal{H}_2(\tau)$, pick $(\tau', \beta_{\tau'})$ from $\Gamma$
           such that $\tau = \tau'$ and compute $\delta \leftarrow \beta_{\tau'} C$
           else $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
           $\sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
      (32) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
      (33) return $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
• $\mathcal{B}$ outputs whatever the experiment would output.
```

We first observe that if $B \neq 0$, it follows that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$ to $\mathcal{A}$ if $c = ab$ and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$ if $c \xleftarrow{\$} \mathbb{Z}_p$. To see this first note the following: (1) the public parameters are distributed identically to both $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$ and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$; (2) random oracle queries are distributed identically to both $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$ and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$ as B is a generator of $\mathbb{G}_1$ and β is sampled uniformly random; (3) the issuance oracle queries are answered identically in the simulated experiment, $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$, and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$; (4) the oracle responses of the request and redemption oracle as well as the re-shuffled challenged tokens generated in the main thread are distributed identically in the simulated experiment, $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$, and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$ for $j \neq t$ (request oracle) or $i \neq t$ (main thread and redemption oracle), which follows from observation (2) and the fact that zero-knowledge proofs are simulated and keys and tokens for $j \neq t$ are computed identically in all three experiments. What remains to show is that if $i = t$ (or $j = t$ in the request oracle) and $c = ab$, the oracle responses of the simulated experiment are distributed identically to $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$ and if $i = t$ (or $j = t$ in the request oracle) and $c \neq ab$, they are distributed identically to $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$. When looking at all queries to the request and redemption oracle with $i = t$ (or $j = t$ in the request oracle) as well as the token potentially generated in the main thread with $i = t$, we observe that the simulated experiments generates outputs $(\mathsf{pk}_t, \pi_{\mathsf{req}}, \{(\delta_\psi, \tau_\psi, (R_\psi^*, \mathsf{pk}_\psi^*, \sigma_\psi^*, \pi_{\mathsf{red}}^\psi))\}_*)$ where the output in the main thread is permuted with the outputs for $i \neq t$. In the tuple, we can ignore the zero-knowledge proofs, the signatures σ_ψ^* as well as the terms R_ψ^* and pk_ψ^* as they are computed exactly the same in the simulated experiment, $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$, and $\mathsf{Exp}_{9,t}^{\mathsf{ul}}$. Now, if $c = ab$ it holds that

$$\begin{aligned}\mathsf{pk}_t &= A = aG_1 \quad (\text{for } a \xleftarrow{\$} \mathbb{Z}_p)\\ \delta_\psi &= \beta_{\tau_\psi} C = (ab\beta_{\tau_\psi})G_1 = a\mathcal{H}_2(\tau_\psi),\end{aligned}$$

which is how the tuple is distributed in $\mathsf{Exp}_{9,t-1}^{\mathsf{ul}}$. To see this, simply replace a by sk_k. If $c \xleftarrow{\$} \mathbb{Z}_p$ such that $c \neq ab$, it holds that

$$\begin{aligned}\mathsf{pk}_t &= A = aG_1 \in_R \mathbb{G}_1\\ \delta_\psi &= \beta_{\tau_\psi} C = (c\frac{b}{b}\beta_{\tau_\psi})G_1 = \frac{c}{b}\mathcal{H}_2(\tau_\psi) \quad (\text{for } c \xleftarrow{\$} \mathbb{Z}_p).\end{aligned}$$

[3]We interpret calls to experiment oracles or inputs to the main thread that cause the oracle or the experiment to query the random oracle as adversarial random oracle queries.

which is how the tuple distributed in $\mathsf{Exp}^{\mathsf{ul}}_{9,t}$. To see this, simply replace $\frac{c}{b}$ by sk_t (we have excluded executions with $b = 0$). As c acts as a one-time pad and is sampled uniformly random, it holds that $\frac{c}{b}$ is distributed uniformly random, what is also expected from sk_t.

Finally, after having shown that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{9,t-1}$ to $\mathcal{A}$ if $(c = ab \wedge b \neq 0)$ and $\mathsf{Exp}^{\mathsf{ul}}_{9,t}$ if $(c \neq ab \wedge b \neq 0)$, we can analyze the output distribution of $\mathcal{B}$. Note that $\mathcal{B}$ outputs 1 if and only if $b \neq 0$ and $\mathcal{A}$ outputs 1 in the simulated experiment. As $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{9,t-1}$ or $\mathsf{Exp}^{\mathsf{ul}}_{9,t}$ depending on whether the received tuple is a real DDH tuple or not, it holds that the output probabilities are

$$\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c = ab] = \frac{p-1}{p} \cdot \Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,t-1}(\lambda) = 1]$$
$$\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c \neq ab] = \frac{p-1}{p} \cdot \Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,t}(\lambda) = 1].$$

and the DDH advantage is

$$\begin{aligned}&\mathsf{Adv}_{\mathsf{ddh}}(\mathcal{B})\\ &= |\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c = ab] - \Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c \neq ab]|\\ &= \frac{p-1}{p} \cdot |\Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,t-1}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,t}(\lambda) = 1]|.\end{aligned}$$

As we bind the advantage of any adversary with a complexity equal to$\mathcal{B}$ by $\mathsf{Adv}_{\mathsf{ddh}}$ due to the DDH assumption, we conclude that

$$\begin{aligned}&|\Pr[\mathsf{Exp}^{\mathsf{ul}}_{8}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,n_{\mathsf{req}}}(\lambda) = 1]|\\ &\leq \frac{n_{\mathsf{req}} \cdot p}{p-1} \cdot \mathsf{Adv}_{\mathsf{ddh}}.\end{aligned}$$

Decouple the re-randomized public keys from the original ones. In another series of $n_{\mathsf{red}} + n$ hybrids, we replace the re-randomized public keys (R^*) generated in the redemption oracle and main thread with randomly sampled group elements. In order to keep track of the number of executed redemption oracle queries, we include a query counter i_{red}. For $t' \in [0, n_{\mathsf{red}} + n]$, we define the series of hybrids as below. Note that $\mathsf{Exp}^{\mathsf{ul}}_{10,0}$ is equal to $\mathsf{Exp}^{\mathsf{ul}}_{9,n_{\mathsf{req}}}$.

Experiment: $\mathsf{Exp}^{\mathsf{ul}}_{10,t'}(\lambda;\mu)$

```
(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\{Q_{\mathsf{redeem}}, C\} \leftarrow \emptyset^2, \{j, i_{\mathsf{red}}, i_Q\} \leftarrow 0^3$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** 0
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{eqs}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** 0
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  **if** $|Q| < n \vee \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ **return** 0
(7)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** 0
(8)  **if** $Q$ contains duplicates **return** 0
(9)  **if** $\exists i \in Q$ s.t. $(i, \cdot, \mathcal{H}_2(\tau)) \in C$ **return** 0
(10) $\forall i \in Q$ :
(11)     $i_Q \leftarrow i_Q + 1, (\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i$
(12)     **if** $i_Q + n_{\mathsf{red}} \leq t'$ $R^* \xleftarrow{\$} \mathbb{G}_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
         **else** $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(13)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(14)     $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(15)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(16)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(17) . . .
. . .
$\underline{O_{\mathsf{Rd}}(i, \tau)}$ :
(28) $i_{\mathsf{red}} \leftarrow i_{\mathsf{red}} + 1$ **if** $i_{\mathsf{red}} > n_{\mathsf{red}}$ **return** $\perp$
(29) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(30) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
(31) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i$,
(32) **if** $i_{\mathsf{red}} \leq t'$ $R^* \xleftarrow{\$} \mathbb{G}_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
     **else** $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(33) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(34) $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(35) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(36) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

We show indistinguishability between $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ and $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ (for $t \in [n_{\mathsf{red}} + n]$) by reducing to the DDH assumption in $\mathbb{G}_1$. More precisely, we build a DDH distinguisher $\mathcal{B}$ that receives a bilinear DDH challenge $\mathsf{ddh} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2, A = aG_1, B = bG_1, C = cG_1)$ with $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2) \leftarrow \mathsf{BGGen}(1^\lambda)$, internally executes an experiment with $\mathcal{A}$, and outputs a bit guess distinguishing real DDH tuples ($c = ab$) from random ones ($c \xleftarrow{\$} \mathbb{Z}_p$). Formally, we define $\mathcal{B}$ as follows:

- $\mathcal{B}$ receives $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2, A, B, C)$ and defines $\mathsf{pp}' \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2)$.
- If $A = 0$ or $B = 0$, $\mathcal{B}$ aborts and outputs 0 (A may not be 0 as r^* is sampled from $\mathbb{Z}_p^*$ and B needs to be a generator of $\mathbb{G}_1$).
- $\mathcal{B}$ executes $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}(\lambda;\mu)$ with randomly sampled random tape μ and the following modifications.
  - $\mathcal{B}$ adapts the experiment's main thread as follows:

```
(1)  $\mathsf{pp} \leftarrow \mathsf{pp}', (Q_{\mathsf{redeem}}, C) \leftarrow \emptyset^2, (i_{\mathsf{red}}, i_Q) \leftarrow 0^2$
(2)  . . .
(9)  $\forall i \in Q$ :
(10)     $i_Q \leftarrow i_Q + 1, (\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i$
(11)     **if** $i_Q + n_{\mathsf{red}} < t$ $R^* \xleftarrow{\$} \mathbb{G}_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
         **elseif** $i_Q + n_{\mathsf{red}} = t$ $R^* \leftarrow A, \mathsf{pk}^* \leftarrow \beta_i C$
         **else** $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \leftarrow r^* \mathsf{pk}_i$
(12)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(13)     $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(14)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(15)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(16) . . .
```

  - $\mathcal{B}$ answers request oracle queries (j, md) as follows:

```
(19) $j \leftarrow j + 1$
(20) $\mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p$
(21) $\beta_j \xleftarrow{\$} \mathbb{Z}_p, \mathsf{pk}_j \leftarrow \beta_j B$
(22) $\pi_{\mathsf{req}} \leftarrow \mathcal{S}_{\mathsf{dlog}}(G_1, \mathsf{pk}_j)$, **return** $(\mathsf{pk}_j, \pi_{\mathsf{req}})$
```

  - $\mathcal{B}$ answers issuance oracle queries as defined in $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$
  - $\mathcal{B}$ answers redemption oracle queries (i, τ) as follows:

```
(28) $i_{\mathsf{red}} \leftarrow i_{\mathsf{red}} + 1$ **if** $i_{\mathsf{red}} > n_{\mathsf{red}}$ **return** $\perp$
(29) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(30) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
(31) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i$
(32) **if** $i_{\mathsf{red}} < t$ $R^* \xleftarrow{\$} \mathbb{G}_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
     **elseif** $i_{\mathsf{red}} = t$ $R^* \xleftarrow{\$} \mathbb{G}_1, \mathsf{pk}^* \leftarrow \beta_i C$
     **else** $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^*, G_1 pk^* \leftarrow r^* \mathsf{pk}_i$
(33) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(34) $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(35) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(36) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

- $\mathcal{B}$ outputs whatever the experiment would output.

We observe that if $A, B \neq 0$ then $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ to $\mathcal{A}$ if $c = ab$ and $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ if $c \neq ab$. To see this first note that the public parameters, the request oracle responses, and the issuance oracle responses in the simulated experiments are distributed identically to both $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ and $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$. When looking at the redemption oracle responses, we observe that all responses for $i_{\mathsf{red}} \neq t$ are distributed identically in the simulated experiment, $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ and $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ as the oracle execution is exactly the same in all three experiments if $i_{\mathsf{red}} \neq t$. If there is a query with $i_{\mathsf{red}} = t$ (i.e., if $t \leq n_{\mathsf{red}}$), the adversary receives a response $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$ with $R^* = A = aG_1$ and $\mathsf{pk}^* = \beta_j C = (\beta_j c)G_1$. If $c = ab$ it holds that $\mathsf{pk}^* = a\mathsf{pk}_j$ which makes the distribution identical to $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$. To see this, simply interpret a as r^*, which is possible as a is distributed uniformly random in $\mathbb{Z}_p^*$ (we have excluded executions with $a = 0$). If $c \neq ab$ it holds that $\mathsf{pk}^* = \frac{c}{b}G_1$, which makes the distribution identical to $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$. Since c is sampled uniformly random, it holds that $\frac{c}{b}G_1$ (we have excluded executions with $b = 0$) is a uniform random group element and, hence, pk^* is uniformly random in $\mathbb{G}_1$ as expected in $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$. The remaining parts of the message, i.e., $(\delta, \tau, \sigma^*, \pi_{\mathsf{red}})$ are computed identically (conditioned on fixed values for R^* and pk^*) in all three experiments and, hence, are distributed identically to $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ or $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ depending on whether $c = ab$ or not. Finally, we look at the tokens generated in the main thread. For each i_Q such that $i_Q + n_{\mathsf{red}} \neq t$ the corresponding token, witness pair (δ_i, ω_i) is computed identically in all three experiments. If there is a $i_Q \in [n]$ such that $i_Q + n_{\mathsf{red}} = t$ (i.e., if $t > n_{\mathsf{red}}$), we observe that the distribution of the generated token-witness pair (δ_i, ω_i) is identically to $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ if $c = ab$ and identically to $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ if $c \neq ab$ based on the same arguments as used for the token-witness pairs returned by the redemption oracle. We note that $i_Q + n_{\mathsf{red}} = t$ and $i_{\mathsf{red}} = t$ contradict each other such that there will only be one token computed based on C. Furthermore, we note that β_j resp. β_i is always defined before being used.

After having established that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}$ to $\mathcal{A}$ if $c = ab \wedge A \neq 0 \wedge B \neq 0$ and $\mathsf{Exp}^{\mathsf{ul}}_{10,t}$ if $c \neq ab \wedge A \neq 0 \wedge B \neq 0$, we can analyze the adversaries success probability analogously to the previous game hop to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{ul}}_{10,t-1}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{10,t}(\lambda) = 1]| \leq (\frac{p}{p-1})^2 \cdot \mathsf{Adv}_{\mathsf{ddh}}$$

and, hence,

$$|\Pr[\mathsf{Exp}^{\mathsf{ul}}_{9,n_{\mathsf{req}}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{10,n_{\mathsf{red}}+n}(\lambda) = 1]| \leq (n_{\mathsf{red}} + n) \cdot (\frac{p}{p-1})^2 \cdot \mathsf{Adv}_{\mathsf{ddh}}.$$

Decouple tokens from each other. Let $n_{\mathcal{H}_2}$ be the number of distinct allowed random oracle queries to $\mathcal{H}_2$[4] (which is lower or equal to the domain of $\mathcal{H}_2$) and n_{req} be the number of allowed queries to the request oracle O_{req}. In another series of $(n_{\mathcal{H}_2} \cdot n_{\mathsf{req}})$ hybrids, we replace the tokens computed from the ℓ-th secret key and the t-th tag-hash by randomly sampled group elements. For $t \in [0, n_{\mathcal{H}_2}]$ and $\ell \in [n_{\mathsf{req}}]$, we define the series of hybrids as below (we first increase t and if $t = n_{\mathcal{H}_2}$ continue with $\ell = \ell + 1$ and $t = 0$). In this hybrid, we explicitly define $\mathcal{H}_2$ to keep track of the order in which the tags are queried and query the oracle for the tag submitted in the main thread to ensure that the tag receives an index. Note that $\mathsf{Exp}^{\mathsf{ul}}_{11,0,1}$ is equal to $\mathsf{Exp}^{\mathsf{ul}}_{10,n_{\mathsf{red}}+n}$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,n_{\mathcal{H}_2},\ell}$ is equal to $\mathsf{Exp}^{\mathsf{ul}}_{11,0,\ell+1}$ for $\ell \in [1, n_{\mathsf{req}} - 1]$.

Experiment: $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}(1^\lambda; \mu)$

```
(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (Q_{\mathsf{redeem}}, C, \Lambda) \leftarrow \emptyset^3, i_\tau \leftarrow 0,\ j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** 0
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{eqs}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** 0
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  $(\cdot) \leftarrow \mathcal{H}_2(\tau)$, let $i^*$ be such that $\tau = \tau_{i^*}$
(7)  **if** $|Q| < n \vee \exists(i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'}$ **return** 0
(8)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** 0
(9)  **if** $Q$ contains duplicates **return** 0
(10) **if** $\exists i \in Q$ s.t. $(i, \cdot, \mathcal{H}_2(\tau)) \in C$ **return** 0
(11) $\forall i \in Q$ :
(12)     $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \xleftarrow{\$} r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
(13)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(14)     **if** $i < \ell \vee (i = \ell \wedge i^* \leq t)$ $r \xleftarrow{\$} \mathbb{Z}_p^*, \delta \leftarrow r\mathcal{H}_2(\tau)$
         **else** $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(15)     $\sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(16)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(17)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(18) ...
...
$\underline{O_{\mathbf{Rd}}(i, \tau)}$ :
(29) **if** $\rho_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(30) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
(31) $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$, let $i^*$ be such that $\tau = \tau_{i^*}$
(32) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(33) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_k, \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(34) **if** $i < \ell \vee (i = \ell \wedge i^* \leq t)$ $r \xleftarrow{\$} \mathbb{Z}_p^*, \delta \leftarrow r\mathcal{H}_2(\tau)$ **else** $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
(35) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(36) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$

$\underline{\mathcal{H}_2(\tau)}$ :
(37) **if** $\exists(\tau^*, O) \in \Lambda$ such that $\tau = \tau^*$ **return** $O$
(38) $i_\tau \leftarrow i_\tau + 1, \tau_{i_\tau} \leftarrow \tau, O \xleftarrow{\$} \mathbb{G}_1, \Lambda \leftarrow \Lambda \cup \{(\tau, O)\}$
(39) **return** $O$
```

We show indistinguishability between $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ (for $t \in [n_{\mathcal{H}_2}]$ and $\ell \in [n_{\mathsf{req}}]$) by reducing to the DDH assumption in $\mathbb{G}_1$. In particular, we build a DDH distinguisher $\mathcal{B}$ that receives a bilinear DDH challenge $\mathsf{ddh} \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, A = aG_1, B = BG_1, C = c_G 1)$ with $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2) \leftarrow \mathsf{BGGen}(1^\lambda)$, internally executes an experiment with $\mathcal{A}$, and outputs a bit guess distinguishing real DDH tuples ($c = ab$) from random ones ($c \neq ab$). The proof is analogous to the one of $\mathsf{Exp}^{\mathsf{ul}}_{9,\cdot}$. Formally, we define $\mathcal{B}$ as follows:

- $\mathcal{B}$ receives $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2, A, B, C)$ and defines $\mathsf{pp}' \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, G_1, G_2)$.
- If B is not a generator of $\mathbb{G}_1$, i.e., if $B = 0$, $\mathcal{B}$ aborts and outputs 0.
- $\mathcal{B}$ executes $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}(\lambda; \mu)$ with randomly sampled random tape μ and the following modifications.
    - $\mathcal{B}$ adapts the random oracle $\mathcal{H}_2$ as follows:

```
(37) **if** $\exists(\tau^*, O) \in \Lambda$ such that $\tau = \tau^*$ **return** $O$
(38) $i_\tau \leftarrow i_\tau + 1$
     **if** $i_\tau < t$ $O \xleftarrow{\$} \mathbb{G}_1$
     **elseif** $i_\tau = t$ $\alpha \xleftarrow{\$} \mathbb{Z}_p, O \leftarrow \alpha G_1$
     **else** $\beta_{i_\tau} \xleftarrow{\$} \mathbb{Z}_p, O \leftarrow \beta_{i_\tau} B$
     $\tau_{i_\tau} \leftarrow \tau, \Lambda \leftarrow \Lambda \cup \{(\tau, O)\}$
```

[4]We interpret calls to experiment oracles or inputs to the main thread that cause the oracle or the experiment to query the random oracle as adversarial random oracle queries.

```
    (39) return $O$
  – $\mathcal{B}$ adapts the experiment's main thread as follows:
    (1) pp $\leftarrow$ pp$'$, $(Q_{\text{redeem}}, C, \Lambda) \leftarrow \emptyset^3, i_\tau \leftarrow 0$
    (2) ...
    (10) $\forall i \in Q$ :
    (11)     $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_i, r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \xleftarrow{\$} r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
    (12)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
    (13)     if $i < \ell \lor (i = \ell \land i^* < t)$ $\delta \xleftarrow{\$} \mathbb{G}_1$
             elseif $i = \ell \land i^* = t$ $\delta \leftarrow \alpha A$
             elseif $i = \ell \land i^* > t$ $\delta \leftarrow \beta_{i^*} C$
             else $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
    (14)     $\sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
    (15)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
    (16)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
    (17) ...
  – $\mathcal{B}$ answers request oracle queries $(j, \mathsf{md})$ as defined in $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$.
  – $\mathcal{B}$ answers issuance oracle queries as defined in $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$.
  – $\mathcal{B}$ answers redemption oracle queries $(i, \tau)$ as follows:
    (29) if $\rho_i \in \{\mathbf{null}, \perp\} \lor (i, \tau) \in Q_{\text{redeem}}$ return $\perp$
    (30) $Q_{\text{redeem}} \leftarrow Q_{\text{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
    (31) $r^* \xleftarrow{\$} \mathbb{Z}_p^*, R^* \leftarrow r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$, let $i^*$ be such that $\tau = \tau_{i^*}$
    (32) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}_{\mathsf{pb}}^{\mathsf{ok}_i})$
    (33) $(\cdot, \mathsf{sk}, \cdot) \leftarrow \rho_k, \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
    (34) if $i < \ell \lor (i = \ell \land i^* < t)$ $\delta \xleftarrow{\$} \mathbb{G}_1$
         elseif $i = \ell \land i^* = t$ $\delta \leftarrow \alpha A$
         elseif $i = \ell \land i^* > t$ $\delta \leftarrow \beta_{i^*} C$
         else $\delta \leftarrow \mathsf{sk} \cdot \mathcal{H}_2(\tau)$
    (35) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
    (36) return $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
• $\mathcal{B}$ outputs whatever the experiment would output.
```

Again, we first observe that if $B \neq 0$, it follows that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ to $\mathcal{A}$ if $c = ab$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ if $c \xleftarrow{\$} \mathbb{Z}_p$. To see this first note the following: (1) the public parameters in the simulated experiment are distributed identically to both $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$; (2) random oracle responses in the simulated experiment are distributed identically to both $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ as B and G_1 are generators of $\mathbb{G}_1$ and β_{i_τ} and α are sampled uniformly random; (3) request and issuance oracle queries are answered identically in the simulated experiment, $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$, and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$; (4) the oracle responses of the redemption oracle for $i \neq \ell \lor (i = \ell \land i^* < t)$ and the token-witness pairs generated in the main thread for $i \neq \ell (i = \ell \land i^* < t)$ are distributed identically in the simulated experiment, $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$, and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$, and (5) all witnesses computed in the redemption oracle and the main thread are distributed identically in the simulated experiment, $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$, and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ and independently from the tokens. Observation (5) follows from the fact that the zero-knowledge proofs are simulated and messages, keys, and signatures are computed identically in all three experiments. What remains to show is that the tokens generated in the simulated experiment's redemption oracle for $i = \ell \land i^* \geq t$ and the simulated experiment's main thread for $i = \ell \land i^* \geq t$ are distributed identically to $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ if $c = ab$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$, otherwise.

For all these tokens it holds that if $c = ab$ (for $a, b \xleftarrow{\$} \mathbb{Z}_p$) then

$$\delta = \alpha A = a\mathcal{H}_2(\tau_{i^*}) \quad (\text{if } i^* = t)$$
$$\delta = \beta_{i^*} C = (ab\beta_{i^*})G_1 = a\mathcal{H}_2(\tau_{i^*}) \quad (\text{if } i^* > t)$$

which is how the tokens are distributed in $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$. To see this, simply substitute a by sk. If $a, b, c \xleftarrow{\$} \mathbb{Z}_p$ such that $c \neq ab$, it holds that

$$\delta = \alpha A = a\mathcal{H}_2(\tau_{i^*}) \quad (\text{if } i^* = t)$$
$$\delta = \beta_{i^*} C = (c\frac{b}{b}\beta_{i^*})G_1 = \frac{c}{b}\mathcal{H}_2(\tau_{i^*}) \quad (\text{if } i^* > t).$$

which is how the tokens are distributed in $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$. To see this, simply substitute a by r and $\frac{c}{b}$ by sk (we have excluded executions with $b = 0$). The variable a is distributed uniformly random in $\mathbb{Z}_p$ what is expected from r. As c acts as a one-time pad and is sampled uniformly random, it holds that $\frac{c}{b}$ is distributed uniformly random in $\mathbb{Z}_p$ exactly as sk.

Finally, after having shown that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ to $\mathcal{A}$ if $(c = ab \land b \neq 0)$ and $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ if $(c \neq ab \land b \neq 0)$, we can analyze the output distribution of $\mathcal{B}$. Note that $\mathcal{B}$ outputs 1 if and only if $b \neq 0$ and $\mathcal{A}$ outputs 1 in the simulated experiment. As $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}$ or $\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}$ depending on whether the received tuple is a real DDH tuple or not, it holds that the output probabilities are

$$\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c = ab] = \frac{p-1}{p} \cdot \Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}(\lambda) = 1]$$
$$\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c \neq ab] = \frac{p-1}{p} \cdot \Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}(\lambda) = 1].$$

and the DDH advantage is

$$\begin{aligned}&\mathsf{Adv}_{\mathsf{ddh}}(\mathcal{B})\\ &= |\Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c = ab] - \Pr[\mathcal{B}(\mathsf{ddh}) = 1 \mid c \neq ab]|\\ &= \frac{p-1}{p} \cdot |\Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}(\lambda) = 1]|.\end{aligned}$$

As we bind the advantage of any adversary with a complexity equal to $\mathcal{B}$ by $\mathsf{Adv}_{\mathsf{ddh}}$ due to the DDH assumption, we conclude that

$$\begin{aligned}&|\Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t-1,\ell}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,t,\ell}(\lambda) = 1]|\\ &\leq \frac{p}{p-1} \cdot \mathsf{Adv}_{\mathsf{ddh}}(\mathcal{B})\end{aligned}$$

and, hence,

$$\begin{aligned}&|\Pr[\mathsf{Exp}^{\mathsf{ul}}_{10,n_{\mathsf{red}}+n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ul}}_{11,n_{\mathcal{H}_2},n_{\mathsf{req}}}(\lambda) = 1]|\\ &\leq \frac{n_{\mathcal{H}_2} \cdot n_{\mathsf{req}} \cdot p}{p-1} \cdot \mathsf{Adv}_{\mathsf{ddh}}.\end{aligned}$$

We note that the adversary can only obtain up to $n_{\mathsf{red}} + n$ tokens, from which at least n need to be from distinct client secret keys sk_j. Therefore, it would suffice to break the relation between just $n_{\mathsf{red}} + 1$ tokens instead of the $n_{\mathcal{H}_2} \cdot n_{\mathsf{req}}$ hybrids we use. Furthermore, for our subsequent security proof, it would be sufficient to break only the relation between the tokens generated in the experiment's main thread and the ones created in the redemption oracle with the same client secret key. All tokens created in the redemption oracle with the same client secret key could still use the same key. Therefore, it would only be necessary to randomize n of the tokens, the ones created in the main thread, and, hence, use only n hybrids.

However, in the experiment, the adversary first sees the possible tag hashes (which, we need to program in the reduction) and some tokens before deciding which tag and client secret keys should be used in the main thread. Therefore, we either have to guess the tokens, that we need to re-randomize before seeing the adversarial selection or re-randomize all potentially queried tokens even if it turns out that they will not be queried. We follow the latter approach. Rewinding the adversary after seeing its selection does not work, as the adversary can adaptively choose its queries and selection based on the query responses, it received before, which will differ after rewinding as we have to program the oracle based on the DDH challenge.

One workaround for this problem would be to use an interactive DDH security assumption, that is structured as follows: The experiment samples a random $a \xleftarrow{\$} \mathbb{Z}_p$ and the adversary queries up to $n_{\mathcal{H}_2}$ random generators B_i and up to n_{red} group elements $C_j = aB_j$. Finally, the adversary selects i^*, such that C_{i^*} has not been queried before, receives C_{i^*}, which is either random or equal to aB_{i^*}, and has to guess whether $C_{i^*} = aB_{i^*}$ or not.

Simplification. In a bridging step, we remove redundant program flow interactions and variable definitions, e.g., the storage and the retrieval of the pre-tokens or the explicit definition of the random oracle $\mathcal{H}_2$. The game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{ul}}_{12}(\lambda; \mu)$

```
(1)  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (Q_{\mathsf{redeem}}, C) \leftarrow \emptyset^2,\ j \leftarrow 0$
(2)  $((\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}), s_0) \leftarrow \mathcal{A}(0, \mathsf{pp})$
(3)  **if** $\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify}(\pi_{\mathsf{pk}}, \mathsf{pk}_{\mathsf{eqs}} = 0)$ **return** $0$
(4)  $\mathsf{sk}_{\mathsf{eqs}} \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{eqs}}(\mu)$, **if** $\mathsf{sk}_{\mathsf{eqs}} = \perp$ **return** $0$
(5)  $(Q, \tau, s_1) \leftarrow \mathcal{A}^{O_{\mathsf{Req}}, O_{\mathsf{Is}}, O_{\mathsf{Rd}}}(1, s_0)$
(6)  **if** $|Q| < n \ \vee\ \exists (i, i') \in Q^2$ s.t. $\mathsf{md}_i \neq \mathsf{md}_{i'} \vee i = i'$ **return** $0$
(7)  **if** $\exists i \in Q$ s.t. $((i, \tau) \in Q_{\mathsf{redeem}} \vee \rho_i \in \{\mathbf{null}, \perp\})$ **return** $0$
(8)  **if** $\exists i \in Q$ s.t. $i \in Q \setminus \{Q\}$
(9)  **if** $\exists i \in Q$ s.t. $(i, \cdot, \mathcal{H}_2(\tau)) \in C$ **return** $0$
(10) $\forall i \in Q$ :
(11)     $r^* \xleftarrow{\$} \mathbb{Z}^*_p, R^* \xleftarrow{\$} r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
(12)     $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(13)     $r \xleftarrow{\$} \mathbb{Z}^*_p, \delta \leftarrow r \cdot \mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(14)     $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(15)     $(\delta_i, \omega_i) \leftarrow (\delta, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
(16) $k \xleftarrow{\$} Q$, pick a random permutation $\phi$ of $Q$
(17) $k^* \leftarrow \mathcal{A}(2, s_1, \phi(k), \{\delta_{\phi(i)}, \omega_{\phi(i)}\}_{i \in Q})$
(18) **if** $k = k^*$ **return** $1$ **else return** $0$

$O_{\mathbf{Req}}(\mathbf{md})$ :

(19) $j \leftarrow j + 1, \mathsf{md}_j \leftarrow \mathsf{md}, \mathsf{sk}_j \xleftarrow{\$} \mathbb{Z}_p, \mathsf{pk}_j \leftarrow \mathsf{sk}_j G_1$
(20) $\pi_{\mathsf{req}} \leftarrow \mathcal{S}_{\mathsf{dlog}}(G_1, \mathsf{pk}_j)$, **return** $(\mathsf{pk}_j, \pi_{\mathsf{req}})$

$O_{\mathbf{Is}}(i, \mathbf{resp})$ :

(21) **if** $\mathsf{pk}_i = \mathbf{null} \vee \rho_i \neq \mathbf{null}$ **return** $\perp$ **else** $\rho_i \leftarrow \perp$
(22) $(R, \mathsf{pk}'_c, X, \sigma, \pi_{\mathsf{is}}) \leftarrow \mathsf{resp}, \mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, \mathcal{H}_1(\mathsf{md}_i) \cdot R, X)$
(23) **if** $\mathsf{ZKP}_{\mathsf{eq\text{-}or}}.\mathsf{Verify}(\pi_{\mathsf{is}}, (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})) = 0$ **return** $0$
(24) $(r, \mathsf{ok}_i) \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{eq\text{-}or}}(\mu)$, **if** $(r, \mathsf{ok}_i) = \perp^2$ **abort** experiment with $0$
(25) **if** $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{eqs}}, \mathsf{msg}, \sigma) = 0 \vee \mathsf{pk}'_c \neq r\mathsf{pk}_i$ **return** $0$
(26) $\rho_i \leftarrow 1$, **return** $1$

$O_{\mathbf{Rd}}(i, \tau)$ :

(27) **if** $\mathsf{ok}_i \in \{\mathbf{null}, \perp\} \vee (i, \tau) \in Q_{\mathsf{redeem}}$ **return** $\perp$
(28) $Q_{\mathsf{redeem}} \leftarrow Q_{\mathsf{redeem}} \cup \{(i, \tau)\}, C \leftarrow C \cup \{(i, \tau, \mathcal{H}_2(\tau))\}$
(29) $r^* \xleftarrow{\$} \mathbb{Z}^*_p, R^* \leftarrow r^* G_1, \mathsf{pk}^* \xleftarrow{\$} \mathbb{G}_1$
(30) $\mathsf{msg}^* \leftarrow (R^*, \mathsf{pk}^*, \mathcal{H}_1(\mathsf{md}_i) \cdot R^*, r^* \mathsf{pk}^{\mathsf{ok}_i}_{\mathsf{pb}})$
(31) $r \xleftarrow{\$} \mathbb{Z}^*_p, \delta \leftarrow r\mathcal{H}_2(\tau), \sigma^* \leftarrow \mathsf{ES.Sign}(\mathsf{sk}_{\mathsf{eqs}}, \mathsf{msg}^*)$
(32) $\pi_{\mathsf{red}} \leftarrow \mathcal{S}_{\mathsf{eq}}(\mathsf{pk}^*, \delta, R^*, \mathcal{H}_2(\tau))$
(33) **return** $(\delta, \tau, (R^*, \mathsf{pk}^*, X^*, \sigma^*, \pi_{\mathsf{red}}))$
```

This game hop is just a bridging step and, hence, does not affect the output distribution. We conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{ul}}_{11, n_{\mathcal{H}_2}, n_{\mathsf{req}}}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{ul}}_{12}(\lambda) = 1].$$

Analysis of final game. In the final game, the only dependence of the token-witness pairs provided to the adversary in the main thread (after shuffling) on the corresponding pre-tokens (selected by the adversary via their index) is the private bit ok_i. However, the private bit can only have one of two values. Hence, we can split the token-witness pairs provided to the adversary into two sets of identically distributed tokens, i.e., the ones generated with $\mathsf{ok}_i = 0$ and the ones with $\mathsf{ok}_i = 1$. Let $Q_0 = \{i \in Q : \mathsf{ok}_i = 0\} \subset Q$, $Q_1 = \{i \in Q : \mathsf{ok}_i = 1\} \subset Q$, $s = |Q|$, $s_0 = |Q_0|$ and $s_1 = |Q_1|$. As the adversary can read the private bit ok_k from the received token-witness pair at position $\phi(k)$ it knows that k needs to be in Q_{ok_k}. However, as all token-witness pairs in Q_{ok_k} are distributed identically and independently of k, the best the adversary can do is guessing an arbitrary element inside Q_{ok_k}.

Given that k is sampled uniformly random from Q it holds that the probability that $\mathsf{ok}_k = b$ is $\frac{s_b}{s}$. As the best the adversary can do is to guess an arbitrary index within Q_{ok_k} and the elements in Q_{ok_k} are shuffled randomly, the success probability of the adversary after observing that $\mathsf{ok}_k = b$ is $\frac{1}{s_b}$. Therefore, the total success probability of the adversary is

$$\frac{s_0}{s} \cdot \frac{1}{s_0} + \frac{s_1}{s} \cdot \frac{1}{s_1} = \frac{2}{s}.$$

Finally, we observe that $s \geq n$ and, hence, that

$$\Pr[\mathsf{Exp}^{\mathsf{ul}}_{13}(\lambda) = 1] \leq \frac{2}{n}$$

By incorporating all game hops, we conclude that

$$\begin{aligned}
&\Pr[\mathsf{Exp}^{\mathsf{ul}}_{\Pi, \mathcal{A}, n}(\lambda) = 1] \leq \\
&\frac{2}{n} + n_{\mathsf{req}} \cdot \Pr[\mathcal{S}_{\mathsf{dlog}} \text{ fails}] \\
&\quad + n_{\mathsf{is}} \cdot \Pr[\mathcal{E}_{\mathsf{eq\text{-}or}} \text{ fails}] \\
&\quad + \Pr[\mathcal{E}_{\mathsf{eqs}} \text{ fails}] \\
&\quad + (n_{\mathsf{red}} + n) \cdot \Pr[\mathcal{S}_{\mathsf{eq}} \text{ fails}] \\
&\quad + \Pr[\text{RO collision with } n_{\mathcal{H}_2} \text{ queries}] \\
&\quad + ((n + n_{\mathsf{red}}) \cdot (\frac{p}{p-1})^2 + (n_{\mathsf{req}} + n_{\mathcal{H}_2} \cdot n_{\mathsf{req}}) \cdot \frac{p}{p-1}) \cdot \mathcal{A}_{\mathsf{ddh}}
\end{aligned}$$

where $n_{\mathsf{req}}, n_{\mathsf{is}}, n_{\mathsf{red}}$ and $n_{\mathcal{H}_2}$ are the number of request oracle, issuance oracle, redemption oracle, and $\mathcal{H}_2$ queries available to the adversary, $\Pr[\text{RO collision with } n_{\mathcal{H}_2} \text{ queries}] = 1 - \frac{p!}{(p - n_{\mathcal{H}_2})! \cdot p^{n_{\mathcal{H}_2}}}$ and $\mathcal{A}_{\mathsf{ddh}}$ is the maximal advantage of a DDH distinguisher running in time approximately equivalent to $\mathsf{Exp}^{\mathsf{ul}}_{\Pi, \mathcal{A}, n}$ plus n_{req} times the runtime of $\mathcal{S}_{\mathsf{dlog}}$, n_{is} times the runtime of $\mathcal{E}_{\mathsf{eq\text{-}or}}$, the runtime of $\mathcal{E}_{\mathsf{eqs}}$ and $(n_{\mathsf{red}} + n)$ times the runtime of $\mathcal{S}_{\mathsf{eq}}$.

B.5 Private Bit Privacy

To show private bit privacy, we prove that for every PPT adversary $\mathcal{A}$ and security parameter λ, it holds that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi, \mathcal{A}, 0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi, \mathcal{A}, 1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

where Π is as defined in Construction 1. We conduct the proof via a sequence of hybrid experiments that are defined as follows:

Inline construction. We start by inlining the concrete computations of Π into the security game (where relevant). The game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{0,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$

```
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ c \leftarrow 0,\ (\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$
(2) $\pi_{\mathsf{pk}} \leftarrow \mathsf{ZKP_{eqs}.Prove}((\mathsf{sk_{eqs}}), (\mathsf{pp}, \mathsf{pk_{eqs}}))$
(3) $\mathsf{sk}^b_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{pk}^b_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} G_1$ for $b \in \{0,1\}$
(4) $\mathsf{pk} \leftarrow (\mathsf{pk_{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}),\ \mathsf{sk} \leftarrow (\mathsf{sk_{eqs}}, \mathsf{sk}^0_{\mathsf{pb}}, \mathsf{sk}^1_{\mathsf{pb}}, \mathsf{pk})$
(5) return $\mathcal{A}^{O_{\mathsf{Is}}, O_{\mathsf{Ch}}, O_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
$O_{\mathsf{Is}}(\mathsf{ok}, \mathsf{md}, \mathsf{req})$ :
    (6) return $\mathsf{Issue}(\mathsf{pp}, \mathsf{req}, \mathsf{md}, \mathsf{ok}, \mathsf{sk})$
$O_{\mathsf{Ch}}(\mathsf{md}, (\mathsf{pk}_c, \pi_{\mathsf{req}}) = \mathsf{req})$ :
    (7) if $c = 1$ return $\perp$ else $c \leftarrow 1$
    (8) if $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\perp$
    (9) $\mathsf{md}^* \leftarrow \mathsf{md},\ m \leftarrow \mathcal{H}_1(\mathsf{md})$
    (10) $v \xleftarrow{\$} \mathbb{Z}^*_p,\ R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c,\ X \leftarrow v \cdot \mathsf{pk}^{\mathsf{ok}^*}_{\mathsf{pb}}$
    (11) $\pi_{\mathsf{is}} \leftarrow \mathsf{ZKP_{eq\text{-}or}.Prove}((v, \mathsf{ok}^*), (R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}))$
    (12) $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X),\ \sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk_{eqs}})$
    (13) return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
$O_{\mathsf{RB}}(\mathcal{P}, \delta, \tau, \mathsf{md}, \omega)$ :
    (14) if $c = 1\ \wedge \mathsf{md} = \mathsf{md}^*$ return $\perp$
    (15) if $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0$ then return $\perp$
    (16) $(R, \cdot, X, \cdot, \cdot) \leftarrow \omega$
    (17) if $X = \mathsf{sk}^1_{\mathsf{pb}} R$ then return 1 else return 0
```

As this is only a change of presentation, it holds that

$$\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{0,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1].$$

Refuse bit at hash collisions. In the first hybrid, we extend the read bit oracle O_{RB} to check whether the submitted metadata md forms a hash collision with the challenged metadata md* (if defined). If this is the case, we do not read the private bit but simply return $\perp$. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$

```
...
$O_{\mathsf{RB}}(\mathcal{P}, \delta, \tau, \mathsf{md}, \omega)$ :
    (14) if $c = 1\ \wedge \mathsf{md} = \mathsf{md}^*$ return $\perp$
    (15) if $c = 1 \wedge \mathcal{H}_1(\mathsf{md}) = \mathcal{H}_1(\mathsf{md}^*)$ return $\perp$
    (16) if $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0$ then return $\perp$
    (17) $(R, \cdot, X, \cdot, \cdot) \leftarrow \omega$
    (18) if $X = \mathsf{sk}^1_{\mathsf{pb}} R$ then return 1 else return 0
```

Equivalently to $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}$ (Section B.3), it follows from the collision resistance of the random oracle and the difference lemma that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{0,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1]| \leq 1 - \frac{p!}{(p - n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}$$

for $n_{\mathcal{H}_1}$ being the maximal number of random oracle queries to $\mathcal{H}_1$ available to $\mathcal{A}$.

Issuance proof simulation. In the second hybrid, we employ simulator $\mathcal{S}_{\mathsf{eq\text{-}or}}$ of proof system $\mathsf{ZKP_{eq\text{-}or}}$ to create the issuance proof π_{is} without making use of the witnesses (v, ok^*). The simulator receives the proof statement $(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$, has the capability to program the random oracle and creates a valid zero-knowledge proof π_{is} except with negligible probability. The hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$

```
...
$O_{\mathsf{Ch}}(\mathsf{md}, (\mathsf{pk}_c, \pi_{\mathsf{req}}) = \mathsf{req})$ :
    (7) if $c = 1$ return $\perp$ else $c \leftarrow 1$
    (8) if $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\perp$
    (9) $\mathsf{md}^* \leftarrow \mathsf{md},\ m \leftarrow \mathcal{H}_1(\mathsf{md})$
    (10) $v \xleftarrow{\$} \mathbb{Z}^*_p,\ R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c,\ X \leftarrow v \cdot \mathsf{pk}^{\mathsf{ok}^*}_{\mathsf{pb}}$
    (11) $\pi_{\mathsf{is}} \leftarrow \mathcal{S}_{\mathsf{eq\text{-}or}}(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
    (12) $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X),\ \sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk_{eqs}})$
    (13) return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
...
```

In $\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}$ the key proof is always valid due to the completeness property of the proof system. Hence, $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ and $\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}$ only differ if $\mathcal{S}_{\mathsf{eq\text{-}or}}$ in $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ is not successful. We use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1]| \leq \Pr[\mathcal{S}_{\mathsf{eq\text{-}or}} \text{ fails}]).$$

for $\mathsf{ok}^* \in \{0,1\}$. Moreover, the complexity of $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ increases by the complexity of $\mathcal{S}_{\mathsf{eq\text{-}or}}$ when compared to $\mathsf{Exp}^{\mathsf{pbp}}_{1,\Pi,\mathcal{A},\mathsf{ok}^*}$.

Simulate signature key proof. Equivalently to $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ (Section B.3), we replace the key proof that is part of the public key pk with a simulated proof computed by $\mathcal{S}_{\mathsf{pk}}$. The hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$

```
(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ c \leftarrow 0,\ (\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^4)$
(2) $\pi_{\mathsf{pk}} \leftarrow \mathcal{S}_{\mathsf{pk}}(\mathsf{pp}, \mathsf{pk_{eqs}})$
(3) ...
```

In $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ the key proof is always valid due to the completeness property of the proof system. Hence, $\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}$ and $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ only differ if $\mathcal{S}_{\mathsf{pk}}$ in $\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}$ is not successful. We use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda) = 1]| \leq \Pr[\mathcal{S}_{\mathsf{pk}} \text{ fails}]).$$

The complexity of $\mathsf{Exp}^{\mathsf{pbp}}_{2,\Pi,\mathcal{A},\mathsf{ok}^*}$ increases by the complexity of $\mathcal{S}_{\mathsf{pk}}$.

Extract the request proof in the challenge oracle. Next, we utilize extractor $\mathcal{E}_{\mathsf{dlog}}$ of proof system $\mathsf{ZKP_{dlog}}$ to extract the witnesses of the zero knowledge proof π_{req} in the challenge oracle after the zero-knowledge proof has been verified correctly. If the extraction fails, i.e. if $\mathsf{sk}_c = \perp$, we return $\perp$. The hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda; \mu)$

```
...
$O_{\mathsf{Ch}}(\mathsf{md}, (\mathsf{pk}_c, \pi_{\mathsf{req}}) = \mathsf{req})$ :
   (7)  if $c = 1$ return $\perp$ else $c \leftarrow 1$
   (8)  if $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\perp$
   (9)  $\mathsf{sk}_c \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{dlog}}(\mu)$, if $\mathsf{sk}_c = \perp$ return $\perp$
  (10)  $\mathsf{md}^* \leftarrow \mathsf{md}$, $m \leftarrow \mathcal{H}_1(\mathsf{md})$
  (11)  $v \xleftarrow{\$} \mathbb{Z}^*_p$, $R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c$, $X \leftarrow v \cdot \mathsf{pk}^{\mathsf{ok}^*}_{\mathsf{pb}}$
  (12)  $\pi_{\mathsf{is}} \leftarrow \mathcal{S}_{\mathsf{eq\text{-}or}}(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
  (13)  $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$, $\sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk_{eqs}})$
  (14)  return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
...
```

The games $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}$ and $\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}$ only differ in the extraction of the zero-knowledge proof and the corresponding success verification. We use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{3,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu) = 1]| \leq \Pr[\mathcal{E}_{\mathsf{dlog}} \text{ fails}].$$

The complexity of $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}$ increases by the complexity of $\mathcal{E}_{\mathsf{dlog}}$.

Random private bit hint if $\mathsf{ok}^* = 0$. In the next hybrid, we randomly sample $X \xleftarrow{\$} \mathbb{G}_1$ in the challenge oracle if $\mathsf{ok}^* = 0$. The hybrid is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu)$

```
...
$\mathcal{O}_{\mathsf{Ch}}$(md, (pk$_c$, $\pi_{\mathsf{req}}$) = req) :
   (7)  if $c = 1$ return $\perp$ else $c \leftarrow 1$
   (8)  if $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\perp$
   (9)  $\mathsf{sk}_c \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{dlog}}(\mu)$, if $\mathsf{sk}_c = \perp$ return $\perp$
  (10)  $\mathsf{md}^* \leftarrow \mathsf{md}$, $m \leftarrow \mathcal{H}_1(\mathsf{md})$
  (11)  $v \xleftarrow{\$} \mathbb{Z}^*_p$, $R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c$
  (12)  if $\mathsf{ok}^* = 1$ $X \leftarrow v \cdot \mathsf{pk}^{\mathsf{ok}^*}_{\mathsf{pb}}$ else $X \xleftarrow{\$} \mathbb{G}_1$
  (13)  $\pi_{\mathsf{is}} \leftarrow \mathcal{S}_{\mathsf{eq\text{-}or}}(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
  (14)  $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$, $\sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk_{eqs}})$
  (15)  return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
...
```

Clearly it holds that

$$\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},1}(\lambda;\mu) = 1] = \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},1}(\lambda;\mu) = 1]$$

as the modification does not affect the experiments with parameter $\mathsf{ok}^* = 1$.

In the experiment with $\mathsf{ok}^* = 0$, we bind the difference in the output distribution between $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$ and $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},0}$ via a reduction to the Decisional Diffie-Hellman problem in $\mathbb{G}_1$. More precisely, we build a DDH-distinguisher $\mathcal{B}$ that utilizes $\mathcal{A}$ to distinguish DDH-tuples as follows:

$\mathcal{B}$ receives $(\mathsf{BG}, A = aG_1, B = bG_1, C)$ with $C = (ab)G_1$ or $C = rG_1$ for random r and simulates an execution of $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}$ or $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$ to $\mathcal{A}$.

- If $B = 0$, $\mathcal{B}$ aborts and outputs 0. (B may not be 0 in the experiment, as v is sampled from $\mathbb{Z}^*_p$.)
- Otherwise, $\mathcal{B}$ executes the experiment as follows:

```
   (1)  $\mathsf{pp} \leftarrow \mathsf{BG}$, $c \leftarrow 0$, $(\mathsf{sk_{eqs}}, \mathsf{pk_{eqs}}) \leftarrow \mathsf{ES.KeyGen}(1^\lambda)$
   (2)  $\pi_{\mathsf{pk}} \leftarrow \mathcal{S}_{\mathsf{pk}}(\mathsf{pp}, \mathsf{pk_{eqs}})$
   (3)  $\mathsf{sk}^1_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}^1_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} G_1$
   (4)  $\mathsf{pk} \leftarrow (\mathsf{pk_{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}} = A, \mathsf{pk}^1_{\mathsf{pb}})$, $\mathsf{sk} \leftarrow (\mathsf{sk_{eqs}}, \perp, \mathsf{sk}^1_{\mathsf{pb}}, \mathsf{pk})$
   (5)  $o \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Is}}, \mathcal{O}_{\mathsf{Ch}}, \mathcal{O}_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
```

- $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{Is}}$ as defined in $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},0}$ (which is equivalent to $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$).
- $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{RB}}$ as defined in $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},\mathsf{ok}^*}$ (which is equivalent to $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$).
- $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{Ch}}$ as follows:

```
   (7)  if $c = 1$ return $\perp$ else $c \leftarrow 1$
   (8)  if $\mathsf{ZKP_{dlog}.Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\perp$
   (9)  $\mathsf{sk}_c \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{dlog}}(\mu)$, if $\mathsf{sk}_c = \perp$ return $\perp$
  (10)  $\mathsf{md}^* \leftarrow \mathsf{md}$, $m \leftarrow \mathcal{H}_1(\mathsf{md})$
  (11)  $R \leftarrow B$, $\mathsf{pk}'_c \leftarrow \mathsf{sk}_c B$, $X \leftarrow C$
  (12)  $\pi_{\mathsf{is}} \leftarrow \mathcal{S}_{\mathsf{eq\text{-}or}}(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
  (13)  $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X)$, $\sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk_{eqs}})$
  (14)  return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
```

- $\mathcal{B}$ outputs o.

First, we observe that $\mathcal{B}$ perfectly simulates experiment $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},0}$ to $\mathcal{A}$ if $B \neq 0 \wedge C = (ab)G_1$ and experiment $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$ if $B \neq 0 \wedge C \neq (ab)G_1$. The changes to the computation of the public parameters and $\mathsf{pk}^0_{\mathsf{pb}}$ do not affect their distribution as BG is computed identically to pp and A is distributed uniformly random, exactly as $\mathsf{pk}^0_{\mathsf{pb}}$ is supposed to. Not knowing $\mathsf{sk}^0_{\mathsf{pb}}$ does not affect the experiment's simulation, as $\mathsf{sk}^0_{\mathsf{pb}}$ is not used in $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},1}$ or $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},1}$ (to read the bit, the oracle only uses $\mathsf{sk}^1_{\mathsf{pb}}$ and assumes the bit to be 0 if it is not 1.) With regard to $T = (R, \mathsf{pk}'_c, X)$, we observe that $T = T_4 = (vG_1, (v\mathsf{sk}_c)G_1, v\mathsf{pk}^0_{\mathsf{pb}})$ in $\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},0}$, $T = T_5 = (vG_1, (v\mathsf{sk}_c)G_1, W)$ for $W \xleftarrow{\$} \mathbb{G}_1$ in $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}$ and $T = T_s = (B, \mathsf{sk}_c B, C)$ for $A = \mathsf{pk}^0_{\mathsf{pb}}$ in the simulated experiment (with $B \neq 0$). It follows that $T_s = T_4$ if $B \neq 0 \wedge C = ab$ and $T_s = T_5$ if $B \neq 0 \wedge C \neq (ab)G_1$. To see this, simply substitute v with b and $\mathsf{pk}^0_{\mathsf{pb}}$ with A.

As b is sampled uniformly random it holds that $\Pr[B = 0] = \frac{1}{p}$. Furthermore, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs in the simulated experiment if $B \neq 0$. Hence, we can conclude

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},0}(\lambda;\mu) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{4,\Pi,\mathcal{A},0}(\lambda;\mu) = 1]| \leq \frac{p}{p-1} \cdot \mathsf{Adv_{ddh}}$$

where $\mathsf{Adv_{ddh}}$ is the advantage in deciding DDH tuples of an adversary running with approximately the same complexity as $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ plus the complexity of $\mathcal{S}_{\mathsf{eq\text{-}or}}$, $\mathcal{S}_{\mathsf{pk}}$ and $\mathcal{E}_{\mathsf{dlog}}$.

Read bit based on $\mathsf{sk}^0_{\mathsf{pb}}$. In the next experiment, we change read bit oracle $\mathcal{O}_{\mathsf{RB}}$ to read the bit based on $\mathsf{sk}^0_{\mathsf{pb}}$ instead of $\mathsf{sk}^1_{\mathsf{pb}}$. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda)$

```
...
$\mathcal{O}_{\mathsf{RB}}$($\mathcal{P}, \delta, \tau$, md, $\omega$) :
  (14)  if $c = 1 \wedge \mathsf{md} = \mathsf{md}^*$ return $\perp$
  (15)  if $c = 1 \wedge \mathcal{H}_1(\mathsf{md}) = \mathcal{H}_1(\mathsf{md}^*)$ return $\perp$
  (16)  if $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0$ then return $\perp$
  (17)  $(R, \cdot, X, \cdot, \cdot) \leftarrow \omega$
  (18)  if $X = \mathsf{sk}^0_{\mathsf{pb}} R$ then return 0 else return 1
```

It is evident that this change only affects the output distribution, if Line 18 is executed with $\mathsf{sk}^0_{\mathsf{pb}} R \neq X \neq \mathsf{sk}^1_{\mathsf{pb}} R$. We call this event F

and bind the probability of event F via a reduction to the unforgeability (EUF-CMA) property of the EQS scheme. More precisely, we build an EQS unforgeability adversary $\mathcal{B}$ that wins the unforgeability game with probability $\Pr[F \text{ in } \mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}]$. $\mathcal{B}$ is defined as follows:

```
$\mathcal{B}$ receives $(\mathsf{pp}', \mathsf{pk}')$, has oracle access to $\mathcal{O}_{\mathsf{Sig}}$ and simulates an execution of $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}$ with adversary $\mathcal{A}$.
• Upon receiving $(\mathsf{pp}', \mathsf{pk}')$, $\mathcal{B}$ executes the experiment as follows:
    (1)  $\mathsf{pp} \leftarrow \mathsf{pp}',\ c \leftarrow 0,\ (\mathsf{sk}_{\mathsf{eqs}}, \mathsf{pk}_{\mathsf{eqs}}) \leftarrow (\bot, \mathsf{pk}')$
    (2)  $\pi_{\mathsf{pk}} \leftarrow \mathcal{S}_{\mathsf{pk}}(\mathsf{pp}, \mathsf{pk}_{\mathsf{eqs}})$
    (3)  $\mathsf{sk}^b_{\mathsf{pb}} \xleftarrow{\$} \mathbb{Z}_p,\ \mathsf{pk}^b_{\mathsf{pb}} \leftarrow \mathsf{sk}^b_{\mathsf{pb}} G_1$ for $b \in \{0,1\}$
    (4)  $\mathsf{pk} \leftarrow (\mathsf{pk}_{\mathsf{eqs}}, \pi_{\mathsf{pk}}, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}}),\ \mathsf{sk} \leftarrow (\mathsf{sk}_{\mathsf{eqs}}, \mathsf{sk}^0_{\mathsf{pb}}, \mathsf{sk}^1_{\mathsf{pb}}, \mathsf{pk})$
    (5)  $o \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Is}}, \mathcal{O}_{\mathsf{Ch}}, \mathcal{O}_{\mathsf{RB}}}(\mathsf{pp}, \mathsf{pk})$
• $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{Is}}$ as defined in $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},0}$ but computes $\sigma$ as
  $\sigma \leftarrow \mathcal{O}_{\mathsf{Sig}}(\mathsf{msg})$ ($\mathsf{sk}_{\mathsf{eqs}}$ is not known.)
• $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{Ch}}$ as defined in $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},0}$ but computes $\sigma$ as
  $\sigma \leftarrow \mathcal{O}_{\mathsf{Sig}}(\mathsf{msg})$ ($\mathsf{sk}_{\mathsf{eqs}}$ is not known.)
• $\mathcal{B}$ answers oracle queries to $\mathcal{O}_{\mathsf{RB}}$ as follows:
   (14)  if $c = 1 \ \wedge \mathsf{md} = \mathsf{md}^*$ return $\bot$
   (15)  if $c = 1 \wedge \mathcal{H}_1(\mathsf{md}) = \mathcal{H}_1(\mathsf{md}^*)$ return $\bot$
   (16)  if $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 0$ then return $\bot$
   (17)  $(R, \mathsf{pk}_c, X, \sigma, \cdot) \leftarrow \omega$
   (18)  if $\mathsf{sk}^0_{\mathsf{pb}} R \neq X \neq \mathsf{sk}^1_{\mathsf{pb}} R$ then
             output forgery $((R, \mathsf{pk}_c, \mathcal{H}_1(\mathsf{md}) \cdot R, X), \sigma)$
   (19)  if $X = \mathsf{sk}^0_{\mathsf{pb}} R$ then return $0$ else return $1$
• If $\mathcal{B}$ did not output anything else before, it outputs $\bot$.
```

First, we observe that $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}$ until $\mathcal{B}$ terminates the simulation (prematurely or not). The changes to the computation of the public parameters, the public key and the signatures do not affect the adversary's view as the EQS unforgeability experiment computes these exactly as $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}$ would. The change in the challenge oracle terminates the simulation when it takes effect and, hence, cannot affect the adversary's view. Hence, we can conclude that event F occurs in the simulated experiment with the same probability as in $\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}$

Next, we show that $\mathcal{B}$ outputs a valid forgery if and only if event F occurs in the simulated experiment. If event F does not occur, the adversary outputs $\bot$ which cannot be a valid forgery. Event F occurs if $\mathcal{O}_{\mathsf{RB}}$ is called with $(\mathcal{P}, \delta, \tau, \mathsf{md}, \omega = (R, \mathsf{pk}_c, X, \sigma, \cdot))$ such that $\mathsf{sk}^0_{\mathsf{pb}} R \neq X \neq \mathsf{sk}^1_{\mathsf{pb}} R$, $m = \mathcal{H}_1(\mathsf{md}) \neq \mathcal{H}_1(\mathsf{md}^*)$, and $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 1$. If event F occurs $\mathcal{B}$ outputs the tuple $(\mathsf{msg} = (R, \mathsf{pk}_c, m \cdot R, X), \sigma)$. As $\mathsf{Verify}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{pk}) = 1$ requires $\mathsf{ES.Verify}(\mathsf{pk}_{\mathsf{ES}}, \mathsf{msg}, \sigma) = 1$, it holds that (msg, σ) is a valid message-signature pair. What remains is to show that msg does not belong to a message class that has been queried by $\mathcal{B}$. For all queries $(R, \cdot, M_3, X)$ submitted by $\mathcal{B}$ it needs to hold that $X = v\mathsf{pk}^0_{\mathsf{pb}} = \mathsf{sk}^0_{\mathsf{pb}} R$ or $X = v\mathsf{pk}^1_{\mathsf{pb}} = \mathsf{sk}^1_{\mathsf{pb}} R$ or $M_3 = \mathcal{H}_1(\mathsf{md}^*) R$. Note that we sample X randomly in the challenge oracle if $\mathsf{ok}^* = 0$. However, in this case it holds that $M_3 = \mathcal{H}_1(\mathsf{md}^*) R$.

For the message $\mathsf{msg} = (R^*, \cdot, M^*_3, X^*)$ returned in event F, it holds that $X^* \neq \mathsf{sk}^0_{\mathsf{pb}} R^*$, $X^* \neq \mathsf{sk}^1_{\mathsf{pb}} R^*$ and $M^*_3 \neq \mathcal{H}_1(\mathsf{md}^*) R^*$. It follows that msg does not belong to a message class that has been queried and, hence, that (msg, σ) is a forgery. We conclude that $\mathcal{B}$ outputs a EQS forgery with probability $\Pr[F \text{ in } \mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}]$ which allows us to bind the probability of F by $\mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$. From the difference lemma it follows that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu) = 1]| \leq \mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$$

where $\mathsf{Adv}_{\mathsf{eqs\text{-}uf}}$ is with respect to an adversary running with approximately the same complexity as $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ plus the complexity of $\mathcal{S}_{\mathsf{eq\text{-}or}}$, $\mathcal{S}_{\mathsf{pk}}$ and $\mathcal{E}_{\mathsf{dlog}}$.

Random private bit hint if $\mathsf{ok}^* = 1$. Analogously to $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},\mathsf{ok}^*}$, we remove the utilization of $\mathsf{sk}^1_{\mathsf{pb}}$ in the challenge oracle by randomly sampling $X \xleftarrow{\$} \mathbb{G}_1$ if $\mathsf{ok}^* = 1$ (in $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},\mathsf{ok}^*}$ we sampled $X \xleftarrow{\$} \mathbb{G}_1$ if $\mathsf{ok}^* = 0$).

```
Experiment: $\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},\mathsf{ok}^*}(\lambda;\mu)$
...
$\mathcal{O}_{\mathsf{Ch}}(\mathsf{md}, (\mathsf{pk}_c, \pi_{\mathsf{req}}) = \mathsf{req})$ :
     (7)  if $c = 1$ return $\bot$ else $c \leftarrow 1$
     (8)  if $\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify}(\pi_{\mathsf{req}}, (G_1, \mathsf{pk}_c)) = 0$ return $\bot$
     (9)  $\mathsf{sk}_c \leftarrow \mathcal{E}^{\mathcal{A}}_{\mathsf{dlog}}(\mu)$, if $\mathsf{sk}_c = \bot$ return $\bot$
    (10)  $\mathsf{md}^* \leftarrow \mathsf{md},\ m \leftarrow \mathcal{H}_1(\mathsf{md})$
    (11)  $v \xleftarrow{\$} \mathbb{Z}^*_p,\ R \leftarrow vG_1, \mathsf{pk}'_c \leftarrow v \cdot \mathsf{pk}_c$
    (12)  $X \xleftarrow{\$} \mathbb{G}_1$
    (13)  $\pi_{\mathsf{is}} \leftarrow \mathcal{S}_{\mathsf{eq\text{-}or}}(R, X, G_1, \mathsf{pk}^0_{\mathsf{pb}}, \mathsf{pk}^1_{\mathsf{pb}})$
    (14)  $\mathsf{msg} \leftarrow (R, \mathsf{pk}'_c, mR, X),\ \sigma \leftarrow \mathsf{ES.Sign}(\mathsf{msg}, \mathsf{sk}_{\mathsf{eqs}})$
    (15)  return $\mathsf{resp} = (R, pk'_c, X, \sigma, \pi_{\mathsf{is}})$
...
```

As this change does not affect the execution with $\mathsf{ok}^* = 0$, we conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},0}(\lambda;\mu) = 1] = \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},0}(\lambda;\mu) = 1].$$

Furthermore, we can show that

$$|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},1}(\lambda;\mu) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{6,\Pi,\mathcal{A},1}(\lambda;\mu) = 1]| \leq \frac{p}{p-1}\mathsf{Adv}_{\mathsf{ddh}}$$

where $\mathsf{Adv}_{\mathsf{ddh}}$ is with respect to an adversary running with complexity approximately equal to the complexity of $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ plus the complexity of $\mathcal{S}_{\mathsf{eq\text{-}or}}$, $\mathcal{S}_{\mathsf{pk}}$ and $\mathcal{E}_{\mathsf{dlog}}$. The reduction works analogously to the one we presented for $\mathsf{Exp}^{\mathsf{pbp}}_{5,\Pi,\mathcal{A},\mathsf{ok}^*}$.

Result of the game hops. The final game does no longer use the bit ok^*. Therefore, it is obvious that

$$\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},0}(\lambda;\mu) = 1] = \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{7,\Pi,\mathcal{A},1}(\lambda;\mu) = 1].$$

By incorporating the differences between all game-hops, we conclude that

$$\begin{aligned}&|\Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},1}(\lambda) = 1]| \leq \\ &2 \cdot ((1 - \frac{p!}{(p - n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}) + \Pr[\mathcal{S}_{\mathsf{eq\text{-}or}} \text{ fails}] + \Pr[\mathcal{S}_{\mathsf{pk}} \text{ fails}] + \\ &\Pr[\mathcal{E}_{\mathsf{dlog}} \text{ fails}] + \frac{p}{p-1} \cdot \mathsf{Adv}_{\mathsf{ddh}} + \mathsf{Adv}_{\mathsf{eqs\text{-}uf}}) \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

where $n_{\mathcal{H}_1}$ is the number of random oracle queries to $\mathcal{H}_1$ available to the adversary and all advantages are with respect to an adversary with complexity that is approximately equal to $\mathsf{Exp}^{\mathsf{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ plus the complexity of $\mathcal{S}_{\mathsf{eq\text{-}or}}$, $\mathcal{S}_{\mathsf{pk}}$, and $\mathcal{E}_{\mathsf{dlog}}$.

C Proof of Theorem 3

In this section, we prove Theorem 3 stating security of our MAC-based construction (cf. Section 5.3). We do so by showing correctness, freshness, unforgeability, unlinkability and private bit privacy.

C.1 Correctness

By the definition of algorithms Setup, SKeyGen, Req, Issue, Final, and Redeem, the completeness of the zero-knowledge proof systems $(\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify})$, $(\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify})$, $(\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Verify})$, and $(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$, it follows that if $\mathsf{sk}_{\mathsf{ok},2} \neq 0$ then

$$M_2^* = (\mathsf{sk}_{\mathsf{ok},1} + \mathsf{sk}_{\mathsf{ok},2} \cdot \mathsf{sk}_c + \mathsf{sk}_{\mathsf{ok},3} \cdot \mathcal{H}_1(\mathsf{md}))M_1^*$$
$$\mathsf{pk}^*_{\mathsf{ok}} = \mathsf{sk}_c M_1^*$$
$$\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify}(\pi_{\mathsf{red}}, (\mathsf{pk}^*_{\mathsf{ok}}, \delta, R^*, \mathcal{H}_2(\tau))) = 1$$

for each $(\delta, \tau, \omega = (M_1^*, M_2, \pi_{\mathsf{red}}))$ produced by the execution of Redeem with the inputs defined as in the correctness definition. As correctness is only defined with respect to $\tau \in \mathcal{P}$ (the initial tag verification cannot fail) and the above conditions are exactly what is verified by ReadBit, we conclude that if $\mathsf{sk}_{\mathsf{ok},2} \neq 0$ then

$$\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) = \mathsf{ok}.$$

As $\mathsf{sk}_{\mathsf{ok},2} \neq 0$ is sampled uniformly random, it holds that $\Pr[\mathsf{sk}_{\mathsf{ok},2} = 0] \leq \mathsf{negl}(\lambda)$. Hence, we conclude that

$$\Pr[\mathsf{ReadBit}(\mathsf{pp}, \mathcal{P}, \delta, \tau, \mathsf{md}, \omega, \mathsf{sk}) \neq \mathsf{ok}] \leq \mathsf{negl}.$$

C.2 Freshness

As tokens are generated exactly the same as in the EQS-based construction, i.e., as $\delta = \mathsf{sk}_c\mathcal{H}_2(\tau)$ for $\mathsf{sk}_c \xleftarrow{\$} \mathbb{Z}_p$, it follows that freshness can be proven equivalently to the EQS-based construction (cf. Section B.2).

C.3 Unforgeability

We present a proof draft showing unforgeability of our MAC-based construction. To do so, we have to show that for every PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}] \leq \mathsf{negl}(\lambda)$$

where Π_{MAC} is the construction defined in Construction 2 (Figure 7) and $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}$ is the adaption of $\mathsf{Exp}^{\mathsf{uf}}_{\Pi,\mathcal{A},n}$ described in Section 3.6.

The proof is based on a series of hybrid experiments. Let $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},0} = \mathsf{Exp}^{\mathsf{ul}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}$. The subsequent hybrids are defined as below. We note that the proof has some overlap with the unforgeability proof of the EQS-based construction (Construction 1, Figure 6) in Section B.3. Hence, we refer the reader to Section B.3 for more details about the game hops that are equivalent.

- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},1}$ we inline the construction into the security game. This change only affects the presentation. The output distribution remains the same.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},2}$, we change oracle $\mathcal{O}_{\mathsf{Is}}$ to keep track of all metadata random oracle responses and abort with output 0 if a collision is detected. This is analogous to $\mathsf{Exp}^{\mathsf{uf}}_{1,\Pi,\mathcal{A},n}$ of Section B.3. The change in the output distribution is equal to the probability of a collision in an ideal hash function on $n_{\mathcal{H}_1}$ queries, i.e., $1 - \frac{p!}{(p-n_{\mathcal{H}_1})!\cdot p^{n_{\mathcal{H}_1}}}$, which is negligible.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},3}$, we simulate the public key proof π_{pk} using the public key proof's simulator $\mathcal{S}_{\mathsf{4DL}}$. This is analogous to $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$ of Section B.3. The change in the output distribution is negligible due to the zero-knowledge property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{4DL}}, \mathsf{ZKP.Verify}_{\mathsf{4DL}})$.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},4}$, we simulate the issuance proofs using the issuance proof system's simulator $\mathcal{S}_{\mathsf{PdOr}}$. The change in the output distribution is negligible due to the zero-knowledge property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{PdOr}}, \mathsf{ZKP.Verify}_{\mathsf{PdOr}})$.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},5}$, we replace the Pedersen commitment of the MAC keys by randomly sampled group elements $C_b \xleftarrow{\$} \mathbb{G}$. This change does not affect the output distribution as the Pedersen commitment is perfectly hiding (the mask u acts as a one-time pad). Note that the issuance zero-knowledge proofs are simulated and, hence, are not affected by this change.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},6}$, we adapt the issuance oracle $\mathcal{O}_{\mathsf{Is}}$ to extract the witness (sk_c) of the request proofs π_{req} via extractor $\mathcal{E}_{\mathsf{dlog}}$ after successfully verifying the proofs. If the extraction fails, we return $\perp$. The change in the output distribution is negligible due to the knowledge soundness property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{dlog}}, \mathsf{ZKP.Verify}_{\mathsf{dlog}})$.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},7}$, we extract both, witness sk_c^* and statement $(\mathsf{pk}^*, \delta, M_1^*, \mathcal{H}_2(\tau))$, from the redemption proofs in the read bit oracle $\mathcal{O}_{\mathsf{RB}}$ using the statement oblivious extractor of the redemption proof system, $\mathcal{E}^{\mathsf{eq}}_{\mathsf{Ob}}$. If the extractor returns $\perp$ or the extracted statement does not match $(\mathsf{pk}^*_b, \delta, M_1^*, \mathcal{H}_2(\tau))$ for any $b \in \{0,1\}$ as computed by the oracle, we return $\perp$. The change in the output distribution is negligible due to the statement-oblivious knowledge soundness property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{eq}}, \mathsf{ZKP.Verify}_{\mathsf{eq}})$.
- In $\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},8}$, we further adapt the read bit oracle $\mathcal{O}_{\mathsf{RB}}$. We don't compute pk^*_b or verify the zero-knowledge proof π_{red}. Instead, we check if the MAC verifies successfully with respect to message $(\mathsf{sk}_c^*, \mathcal{H}_1(\mathsf{md}))$ under one of the MAC keys by computing $\mathsf{ok}_b \leftarrow M_2^* = (\mathsf{sk}^s_{b,1} + \mathsf{sk}^s_{b,2} \cdot \mathsf{pk}^*_c + \mathsf{sk}^s_{b,3} \cdot \mathcal{H}_1(\mathsf{md})) \cdot M_1^*$ for $b \in \{0,1\}$. If this is the case, we return the private bit associated with the MAC key under which the MAC verified successfully. This change does not affect the output distribution as the success conditions are equivalent – note that the success of extractor $\mathcal{E}^{\mathsf{eq}}_{\mathsf{Ob}}$, which is signaled by a non-$\perp$ output, guarantees successful verification of the zero-knowledge proof under the extracted statement.
- In the final experiment, we can show that $\Pr[\mathsf{Exp}^{\mathsf{uf\text{-}pv}}_{\mathsf{MAC},9}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ via a reduction to the MAC unforgeability property as stated by Theorem 1. Let $\mathcal{B}$ be the adversary

attacking the MAC unforgeability game $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$ by internally simulating an execution of $\mathsf{Exp}^{\text{uf-pv}}_{\mathsf{MAC},9}$ with adversary $\mathcal{A}$. When generating the issuer public key and when issuing tokens, $\mathcal{B}$ cannot know the private bit ok^* that is returned by $\mathcal{A}$ after being executed. Hence, $\mathcal{B}$ does not know if the adversary's response will allow it to extract a MAC forgery (if any) under MAC key $(\mathsf{sk}^s_{0,1}, \mathsf{sk}^s_{0,2}, \mathsf{sk}^s_{0,3})$ or $(\mathsf{sk}^s_{1,1}, \mathsf{sk}^s_{1,2}, \mathsf{sk}^s_{1,3})$. To circumvent this problem, $\mathcal{B}$ guesses a bit $\mathsf{ok_g}$ and runs the reduction assuming that $\mathsf{ok}^* = \mathsf{ok_g}$. This means that $\mathcal{B}$ samples $(\mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},1}, \mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},2}, \mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},3})$ for $\bar{\mathsf{ok}}_\mathsf{g} = 1 - \mathsf{ok_g}$ itself. The public key components for $b = \bar{\mathsf{ok}}_\mathsf{g}$ and issuance query responses for $\mathsf{ok} = \bar{\mathsf{ok}}_\mathsf{g}$ are computed based on $(\mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},1}, \mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},2}, \mathsf{sk}^s_{\bar{\mathsf{ok}}_\mathsf{g},3})$. The public key components for $b = \mathsf{ok_g}$ are set to be the public key components received by $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$ and issuance queries with $\mathsf{ok} = \mathsf{ok_g}$ are answered by querying a MAC on the extracted client secret key and the metadata hash from the MAC oracle of $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$. Queries to the read bit oracle $\mathcal{O}_{\mathsf{RB}}$ can be answered based on the MAC verification oracle of $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$. This is due to the fact that we changed the read bit oracle to extract the statement and the witness from the redemption proof, first, and then verify the MAC. The remaining steps of the reduction are equivalent to the reduction to the signature unforgeability property of $\mathsf{Exp}^{\mathsf{uf}}_{n+2,\Pi,\mathcal{A},n}$.
  As the secret MAC key sampled by $\mathcal{B}$ and the one sampled by $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$ are distributed identically, it follows that the public keys and MACs computed by $\mathcal{B}$ itself and the ones sampled by $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$ are distributed identically as well. Hence, it is not possible for $\mathcal{A}$ to derive any information about $\mathsf{ok_g}$. It follows that $\mathcal{B}$ guesses ok^* with probability $\frac{1}{2}$. We conclude that $\mathcal{B}$ successfully attacks $\mathsf{Exp}^{\mathsf{MAC}}_{\mathcal{A}}$ with probability equal to $\frac{1}{2} \cdot \Pr[\mathsf{Exp}^{\text{uf-pv}}_{\mathsf{MAC},9}(\lambda) = 1]$, which implies that $\Pr[\mathsf{Exp}^{\text{uf-pv}}_{\mathsf{MAC},9}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.
- As there is a polynomial number of game hops and any game-hop introduces at most a negligible difference in the output distribution, it holds that $\Pr[\mathsf{Exp}^{\text{uf-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

C.4 Unlinkability

We present a proof draft showing unlinkability, in particular 2-unlikability of our MAC-based construction. To do so, we have to show that for every PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\mathsf{Exp}^{\text{ul-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}(\lambda) = 1] \leq \frac{2}{n} + \mathsf{negl}(\lambda)$$

where Π_{MAC} is the construction defined in Construction 2 (Figure 7). and $\mathsf{Exp}^{\text{ul-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}$ is the adaption of $\mathsf{Exp}^{\mathsf{ul}}_{\Pi,\mathcal{A},n}$ described in Section 3.6.

The proof is based on a series of hybrid experiments. Let $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},0} = \mathsf{Exp}^{\text{ul-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},n}$. The subsequent hybrids are defined as below. We note that the proof is analogous to the 2-unlinkability proof of the EQS-based construction (Construction 1, Figure 6) in Section B.4. Most of the game hops and reductions are equivalent. Hence, we refer the reader to Section B.4 for more details about the game hops applied in this proof.

- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},1}$, we inline the construction into the security game. We keep track of the metadata and the client key pair via variables md_i and $(\mathsf{sk}_{(c,i)}, \mathsf{pk}_{(c,i)})$. This change only affects the presentation. The output distribution remains the same.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},2}$, we simulate the request proofs with the request proof simulator $\mathcal{S}_{\mathsf{dlog}}$. This is analogous to $\mathsf{Exp}^{\mathsf{ul}}_2$ of Section B.4. The change in the output distribution is negligible due to the zero-knowledge property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{dlog}}, \mathsf{ZKP.Verify}_{\mathsf{dlog}})$.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},3}$, we extract the witnesses $(\mathsf{sk}^s_{\mathsf{ok}_i,1}, u_{\mathsf{ok}_i}, v_i, \mathsf{ok}_i)$ of the issuance proofs π_{is} via extractor $\mathcal{E}_{\mathsf{PdOr}}$. This is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible due to the knowledge soundness property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{PdOr}}, \mathsf{ZKP.Verify}_{\mathsf{PdOr}})$.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},4}$, we extract the witnesses $(\mathsf{sk}^s_{0,2}, \mathsf{sk}^s_{0,3}, \mathsf{sk}^s_{1,2}, \mathsf{sk}^s_{1,3},)$ of the public key proof π_{pk} via extractor $\mathcal{E}_{\mathsf{4DL}}$. This is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible due to the knowledge soundness property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{4DL}}, \mathsf{ZKP.Verify}_{\mathsf{4DL}})$.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},5}$, we simulate the redemption proofs with the redemption proof system's simulator $\mathcal{S}_{\mathsf{eq}}$. This is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible due to the zero-knowledge property of the proof system $(\mathsf{ZKP.Prove}_{\mathsf{eq}}, \mathsf{ZKP.Verify}_{\mathsf{eq}})$.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},6}$, we check for each extracted witness of an issuance proof if it depicts a violation of the commitment's binding property. More precisely, when extracting a witness $(\mathsf{sk}^s_{0,1}, u_0, \cdot, 0)$ or a witness $(\mathsf{sk}^s_{1,1}, u_1, \cdot, 1)$ for the first time, we store $(\hat{\mathsf{sk}}^s_{0,1}, \hat{u}_0) = (\mathsf{sk}^s_{0,1}, u_0)$ or $(\hat{\mathsf{sk}}^s_{1,1}, \hat{u}_1) = (\mathsf{sk}^s_{1,1}, u_1)$, respectively. When extracting a witness $(\mathsf{sk}^s_{b,1}, u_b, \cdot, b)$ while $(\hat{\mathsf{sk}}^s_{b,1}, \hat{u}_b)$ is already defined, we check whether $(\mathsf{sk}^s_{b,1}, u_b) = (\hat{\mathsf{sk}}^s_{b,1}, \hat{u}_b)$. If this is not the case, we abort the experiment with output 0. The change in the output distribution is negligible since the Pedersen commitment used to compute C_0 and C_1 is computational binding.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\mathsf{MAC},7}$, we do not re-randomize the MAC when redeeming a token (in the redemption oracle or the experiment's main thread) but compute them from scratch with the extracted secret keys, i.e., $r^* \xleftarrow{\$} \mathbb{Z}^*_p, M^*_1 \leftarrow r^* G, \mathsf{pk}^* \leftarrow r^* \cdot \mathsf{pk}_{(c,i)}, M^*_2 \leftarrow (\hat{\mathsf{sk}}^s_{\mathsf{ok}_i,1} + \mathcal{H}_1(\mathsf{md}_i) \cdot \mathsf{sk}^s_{\mathsf{ok}_i,3}) M^*_1 + (r \cdot \mathsf{sk}^s_{\mathsf{ok}_i,2}) \mathsf{pk}^*$.
  The server keys are guaranteed to be available as we only redeem tokens, for which we received a valid issuance message and, hence, a valid issuance proof, before. The same holds for ok_i. Hence, it is guaranteed that the experiment is in possession of all secret data associated with redeemed tokens, i.e., $(\mathsf{sk}^s_{\mathsf{ok}_i,1}, \mathsf{sk}^s_{\mathsf{ok}_i,2}, \mathsf{sk}^s_{\mathsf{ok}_i,3}, \mathsf{ok}_i)$.
  Note that we do not need $\mathsf{sk}_{(c,i)}$ to compute (pk^*, M^*_1, M^*_2).

This is necessary for later reductions to the DDH assumption. The output distribution is the same as in $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},6}$ as both experiments compute the same token witnesses (pk^*, M_1^*, M_2^*).

- In $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},8}$, we abort the experiment with output 0, if the hash of the tag submitted by the adversary in the experiment's main thread collides with the hash of any other tag queried by the adversary or the experiment. This is analogous to the unlinkability proof of the EQS-based construction. Let $n_{\mathcal{H}_2}$ be the number of allowed oracle queries . The change in the output distribution is equal to the probability of a collision in an ideal hash function on $n_{\mathcal{H}_2}$ queries, i.e., $1 - \frac{p!}{(p-n_{\mathcal{H}_2})! \cdot p^{n_{\mathcal{H}_2}}}$, which is negligible.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},9}$, we break the correlation between tokens and MACs. We do so by randomly sampling the client public keys ($\mathsf{pk}_c \xleftarrow{\$} \mathbb{G}$) instead of computing them based on the client secret key. The MACs in the redemption oracle and the main thread are now computed based on the client's randomly sampled public key but the tokens based on the client's secret key (which is now independent of the public key). This modification is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible, which can be shown via a reduction to the DDH assumption.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},10}$, we break the correlation between the client public keys given to the adversary via the request oracle and the randomized public keys used in the computation of the MACs in the redemption oracle and the experiment's main thread. We do so by randomly sampling the re-randomized public key, i.e., $\mathsf{pk}^* \xleftarrow{\$} \mathbb{G}$, instead of computing it based on the original client public key, i.e., $\mathsf{pk}^* \leftarrow r^* \cdot \mathsf{pk}_{(c,i)}$. This modification is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible, which can be shown via a reduction to the DDH assumption.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},11}$, we break the correlation between the tokens computed with the same secret client key by sampling tokens as random group elements. To be more precise, whenever a token δ on tag τ is generated, we sample some $r_\delta \xleftarrow{\$} \mathbb{Z}_p$ and define $\delta \leftarrow r_\delta \cdot \mathcal{H}_2(\tau)$ instead of $\delta \leftarrow \mathsf{sk}_{(c,i)}\mathcal{H}_2(\tau)$. We cannot sample δ directly from $\mathbb{G}$ as we need to ensure that a tag τ' with $\mathcal{H}_2(\tau') = 0$ causes all tokens computed based on τ' to be 0. This modification is analogous to the unlinkability proof of the EQS-based construction. The change in the output distribution is negligible, which can be shown via a reduction to the DDH assumption.
- In $\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},11}$, the only dependence on the pre-tokens of the tokens generated and shuffled in the experiment's main thread is the private bit. However, the private bit can only have one of two values. Hence, we can split the generated tokens into two sets of identically distributed tokens. The best the adversary can do, is to read the private bit ok^* of the challenged token δ^*, the one at position $\phi(k)$, and arbitrarily guess any of the pre-tokes (referenced via their index) in set Q, that has private bit ok^*. Let $s = |Q|$ be the number of pre-tokens selected by the adversary (via their indices), let s_0 be the number of pre-tokens with private bit 0 and s_1 the number of pre-tokens with private bit 1. Since the experiment samples k uniformly random from the s indices and the best the adversary can do is to pick an arbitrary token that has the same private bit as the challenged token δ^*, it holds that the adversary wins with probability $\frac{s_0}{s} \cdot \frac{1}{s_0} + \frac{s_1}{s} \cdot \frac{1}{s_1} = \frac{2}{s}$. As $s \leq n$, we can conclude that $\Pr[\mathsf{Exp}^{\text{ul-pv}}_{\text{MAC},11}(\lambda) = 1] \leq \frac{2}{n}$. As there is a polynomial number of game hops and any game-hop introduces at most a negligible difference in the output distribution, it holds that $\Pr[\mathsf{Exp}^{\text{ul-pv}}_{\Pi_{\text{MAC}},\mathcal{A},n}(\lambda) = 1] \leq \frac{2}{n} + \mathsf{negl}(\lambda)$.

C.5 Private Bit Privacy

We present a proof draft showing private bit privacy of our MAC-based construction. To do so, we have to show that for every PPT adversary $\mathcal{A}$ it holds that

$$|\Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\text{MAC}},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\text{MAC}},\mathcal{A},1}(\lambda) = 0] \leq \mathsf{negl}(\lambda)$$

where Π_{MAC} is the construction defined in Construction 2 (Figure 7) and $\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\text{MAC}},\mathcal{A},\mathsf{ok}^*}$ is the adaption of $\mathsf{Exp}^{\text{pbp}}_{\Pi,\mathcal{A},\mathsf{ok}^*}$ described in Section 3.6. Most importantly, $\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\text{MAC}},\mathcal{A},\mathsf{ok}^*}$ includes a verification oracle $\mathcal{O}_{\mathsf{Vf}}$ that takes the same inputs as the read-bit oracle $\mathcal{O}_{\mathsf{RB}}$, runs the read-bit algorithm $\mathsf{ReadBit}(\ldots)$ (without first checking the metadata or the flag c), and returns 0 if $\mathsf{ReadBit}(\ldots) = \bot$ and 1 otherwise.

Before presenting the proof, we recall two basic tools, the generic group model and the Schwartz-Zippel lemma. We note that the use of the generic group model in this proof does not affect the overall security as the unforgeability of the underlying MAC construction has only be proven in the generic group model as well [12].

The generic group model. We make use of the generic group model (GGM) [36]. In the generic group model, the adversary does not observe the group elements it operates on, but only random labels representing them. The adversary performs computations on group elements through a black-box oracle, which executes the operations without disclosing the resulting group element – the adversary receives just the label of the resulting element. Group elements returned by the adversary can, hence, be interpreted as multi-variate polynomials, where the indeterminates represent the group elements provided to the adversary. The coefficients of the polynomials are decided by the adversary while the allowed combinations of indeterminates are decided by the protocol. This is due to the fact that the adversary can only add group elements while the experiment might know the discrete logarithm of some of the group elements and, hence, can apply the discrete logarithm of one group element to another. The indeterminates are described by the discrete logarithm of the group elements to a common basis, which allows the multiplication of group elements and the application of the Schwartz-Zippel lemma. When applying the Schwartz-Zippel lemma, we replace the indeterminates describing group elements by the respective indeterminates describing the discrete logarithm of the group elements to the common basis.

The Schwartz–Zippel lemma. We utilize the Schwartz–Zippel lemma [35, 38], that is defined as follows:

THEOREM 4. *Let $F \in R[\bar{x}_1, \ldots \bar{x}_n]$ be a non-zero polynomial of degree $d \leq 0$ over an integral domain R. Let S be a finite subset of R and let $(x_1, \ldots, x_n)$ be selected at random independently and uniformly from S. Then,*

$$\Pr[F(x_1, \ldots, x_n) = 0] \leq \frac{d}{|S|}.$$

Here, we are only interested in the case that $R = S = \mathbb{Z}_p$.

The security proof. The proof is based on a series of hybrid experiments. We note that the proof has some overlap with the private bit privacy proof of the EQS-based construction (Construction 1, Figure 6) in Section B.4. Hence, we refer the reader to Section B.5 for more details about the game hops that are equivalent.

- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},1,\mathsf{ok}^*}$, we return $\perp$ if the hash of the metadata submitted by the adversary to the read bit oracle $\mathcal{O}_{\mathsf{RB}}$ collides with the hash of the challenged metadata md* (if defined). This is analogous to the private bit privacy proof of the EQS-based construction. Let $n_{\mathcal{H}_1}$ be the number of allowed oracle queries. The change in the output distribution is equal to the probability of a collision in an ideal hash function on $n_{\mathcal{H}_1}$ queries, i.e., $1 - \frac{p!}{(p-n_{\mathcal{H}_1})! \cdot p^{n_{\mathcal{H}_1}}}$, which is negligible.
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},2,\mathsf{ok}^*}$, we simulate the issuance proofs with the issuance proof system's simulator $\mathcal{S}_{\mathsf{PdOr}}$. This is analogous to the private bit privacy proof of the EQS-based construction. The change in the output distribution is negligible due to the zero-knowledge property of the proof system ($\mathsf{ZKP.Prove}_{\mathsf{PdOr}}$, $\mathsf{ZKP.Verify}_{\mathsf{PdOr}}$).
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},3,\mathsf{ok}^*}$, we simulate the public key proofs with the public key proof's simulator $\mathcal{S}_{\mathsf{4DL}}$. This is analogous to the private bit privacy proof of the EQS-based construction. The change in the output distribution is negligible due to the zero-knowledge property of the proof system ($\mathsf{ZKP.Prove}_{\mathsf{4DL}}$, $\mathsf{ZKP.Verify}_{\mathsf{4DL}}$).
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},4,\mathsf{ok}^*}$, we extract the witness (sk_c) of all the request proofs π_{req} via extractor $\mathcal{E}_{\mathsf{dlog}}$. While the private bit privacy proof of the EQS-based construction only extracted the request proof witness in the challenge oracle $\mathcal{O}_{\mathsf{Ch}}$, we also extract the request proof's witness from the calls to the issuance oracle $\mathcal{O}_{\mathsf{Is}}$ in this proof. The change in the output distribution is negligible due to the knowledge soundness property of the proof system ($\mathsf{ZKP.Prove}_{\mathsf{dlog}}$, $\mathsf{ZKP.Verify}_{\mathsf{dlog}}$).
- From here on, we diverge from the private bit privacy proof of the EQS-based construction. In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},5,\mathsf{ok}^*}$, we start maintaining sets $\Gamma_{\mathsf{sk}_c,m,\mathsf{ok}}$ that are initially set to $\emptyset$ and populated to contain all MACs issued on messages sk_c and $m = \mathcal{H}_1(\mathsf{md})$ with the private key for private bit ok. Furthermore, we compute the MACs based on the extracted secret keys sk_c instead of the public key. This change does not affect the output distribution.
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},6,\mathsf{ok}^*}$, we replace the Pedersen commitment of the MAC keys by randomly sampled group elements. This change does not affect the output distribution as the Pedersen commitment is perfectly hiding (the mask u acts as a one-time pad). Note that the issuance zero-knowledge proofs are simulated and, hence, are not affected by this change.
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},7,\mathsf{ok}^*}$, we extract both, witness sk^*_c and statement $(\mathsf{pk}^*, \delta, M^*_1, \mathcal{H}_2(\tau))$, from the redemption proofs in the read bit oracle $\mathcal{O}_{\mathsf{RB}}$ and the verify oracle $\mathcal{O}_{\mathsf{Vf}}$ using the statement oblivious extractor of the redemption proof system, $\mathcal{E}^{\mathsf{eq}}_{\mathsf{Ob}}$. If the extractor returns $\perp$ or the extracted statement does not match $(\mathsf{pk}^*_b, \delta, M^*_1, \mathcal{H}_2(\tau))$ for any $b \in \{0,1\}$ as computed by the oracle, we return $\perp$. The change in the output distribution is negligible due to the statement oblivious knowledge soundness property of the proof system ($\mathsf{ZKP.Prove}_{\mathsf{eq}}$, $\mathsf{ZKP.Verify}_{\mathsf{eq}}$).
- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},8,\mathsf{ok}^*}$, we further adapt the read bit oracle $\mathcal{O}_{\mathsf{RB}}$ and the verify oracle $\mathcal{O}_{\mathsf{Vf}}$. We don't compute pk^*_b or verify the zero-knowledge proof π_{red}. Instead, we check if the MAC verifies successfully with respect to message $(\mathsf{sk}^*_c, \mathcal{H}_1(\mathsf{md}))$ under one of the MAC keys. If this is the case, we return the private bit associated with the MAC key under which the MAC verified successfully. This change does not affect the output distribution as the success conditions are equivalent – note that the success of extractor $\mathcal{E}^{\mathsf{eq}}_{\mathsf{Ob}}$, which is signaled by a non-$\perp$ output, guarantees successful verification of the zero-knowledge proof under the extracted statement.
- From here on, we will start making explicit use of the generic group model. In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},9,\mathsf{ok}^*}$, we change the way the read oracle $\mathcal{O}_{\mathsf{RB}}$ and the verification oracle $\mathcal{O}_{\mathsf{Vf}}$ verify the MACs. Let (M^*_1, M^*_2) be the MAC provided by the adversary to the oracle, let sk_c be the secret key extracted from the redemption proof, and let $m = \mathcal{H}_1(\mathsf{md})$. Instead of verifying the MAC, the oracle checks if the MAC forms a linear combination of the MACs $\{M^1_i, M^2_i\} \in \Gamma_{\mathsf{sk}_c,m,b}$ for any $b \in \{0,1\}$, i.e., all MACs on (sk_c, m) under the secret key associated with private bit b. If this is the case, the adversary interprets the MAC to be valid with respect to private bit b.
  As it is infeasible to check whether (M^*_1, M^*_2) forms any linear combinations of the MACs $\{M^1_i, M^2_i\} \in \Gamma_{\mathsf{sk}_c,m,b}$ (for $b \in \{0,1\}$) in the standard model, we use the generic group model to check whether the adversary computed (M^*_1, M^*_2) as a linear combination of the MACs in $\Gamma_{\mathsf{sk}_c,m,b}$. Let F_1 be the polynomial representing M^*_1 and F_2 be the polynomial representing M^*_2. To check, whether (M^*_1, M^*_2) forms a linear combination of the MACs $\{M^1_i, M^2_i\} \in \Gamma_{\mathsf{sk}_c,m,b}$, we check whether $F_1 = \sum_{(M^i_1,\cdot) \in \Gamma_{\mathsf{sk}_c,m,b}} a_i \cdot \bar{M}^i_1$ and $F_2 = \sum_{(\cdot,M^i_2) \in \Gamma_{\mathsf{sk}_c,m,b}} a_i \cdot \bar{M}^i_2$ where a_i are integer coefficients and $\bar{M}^i_1$ and $\bar{M}^i_2$ are the indeterminates representing the discrete logarithm of the MACs provided to the adversary. Importantly, the integers a_i need to be the same in both polynomials and all other coefficients in the polynomials, e.g., for indeterminates $\bar{G}, \bar{H}, \ldots$, are 0.
  Next, we argue why this change does not impact the output distribution, except with a negligible probability. First, we

note that if we interpret a MAC to be valid with respect to bit b in $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},9,\mathsf{ok}^*}$, it holds that (M_1^*, M_2^*) is a valid MAC on message (sk_c, m) under the secret key associated with bit b and, hence, $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},8,\mathsf{ok}^*}$ would also accept the MAC with respect to bit b. This is due to the fact that for each $(M_i^1, M_i^2) \in \Gamma_{\mathsf{sk}_c,m,b}$, it holds that $M_i^2 = (\mathsf{sk}^s_{b,1} + (\mathsf{sk}_c \cdot \mathsf{sk}^s_{b,2}) + (m \cdot \mathsf{sk}^s_{b,3}))M_1$ and, hence, this relation also holds for (M_1^*, M_2^*), if (M_1^*, M_2^*) is indeed a linear combination of the MACs in $\Gamma_{\mathsf{sk}_c,m,b}$. Second, we show that the probability that the adversary provides a token-witness pair to the read oracle $\mathcal{O}_{\mathsf{RB}}$ or the verify oracle $\mathcal{O}_{\mathsf{Vf}}$ in $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},8,\mathsf{ok}^*}$ such that the MAC does not form a linear combination of $\Gamma_{\mathsf{sk}_c,m,b}$ (for $b \in \{0,1\}$) and the oracle still verifies the MAC successfully is negligible. Let ok be the bit associated with the MAC key under which $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},8,\mathsf{ok}^*}$ successfully verifies the MAC, let F_1 be the polynomial representing M_1^* and F_2 be the polynomial representing M_2^*. As the MAC verifies successfully under the secret key associated with bit ok, it needs to hold that $F_2 = F_1 \cdot (\bar{\mathsf{sk}}_{\mathsf{ok},1} + \bar{\mathsf{sk}}_{\mathsf{ok},2} \cdot \mathsf{sk}_c + \bar{\mathsf{sk}}_{\mathsf{ok},3} \cdot m)$ which is equivalent to the condition that $F_* = F_2 - F_1 \cdot (\bar{\mathsf{sk}}_{\mathsf{ok},1} + \bar{\mathsf{sk}}_{\mathsf{ok},2} \cdot \mathsf{sk}_c + \bar{\mathsf{sk}}_{\mathsf{ok},3} \cdot m) = 0$. The polynomial F_* is a 0-polynomial if F_2 and F_1 are 0 polynomials or if (M_1^*, M_2^*) form a linear combination of the MACs in $\Gamma_{\mathsf{sk}_c,m,\mathsf{ok}}$. The former causes the MAC verification to fail – M_1^* may not be 0 in the MAC verification. The latter contradicts the failure event as we are looking for the event that the MAC does not form a linear combination of the MACs in $\Gamma_{\mathsf{sk}_c,m,\mathsf{ok}}$ and is still accepted. As the indeterminates in F^* represent randomly sampled variables (we substitute indeterminates for M_i^2, i.e., $\bar{M}_i^2$, by $(\bar{\mathsf{sk}}_{\mathsf{ok}_i,1} + \bar{\mathsf{sk}}_{\mathsf{ok}_i,2} \cdot \mathsf{sk}_c^i + \bar{\mathsf{sk}}_{\mathsf{ok}_i,3} \cdot h) \cdot \bar{M}_i^1$, which is composed of indeterminates representing randomly sampled variables), and F^* is a non-0 polynomial of degree 3, we can apply the Schwarz-Zippel lemma to conclude that the probability that the adversary finds an accepting MAC (M_1^*, M_2^*), that does not form a linear combination of the MACs in $\Gamma_{\mathsf{sk}_c,m,b}$ is negligible. Hence, we have shown that the change in the output distribution introduced by the new experiment is negligible.

- In $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},10}$, $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},11}$, and $\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},12}$, we replace the MAC components of the challenged MAC with randomly sampled group elements. Recall that the challenged MAC is computed as $M_1 \leftarrow vG$ and $M_2 \leftarrow \mathsf{sk}^s_{\mathsf{ok},1}M_1 + (\mathsf{sk}^s_{\mathsf{ok},2} \cdot \mathsf{sk}_c)M_1 + (\mathsf{sk}^s_{\mathsf{ok},2} \cdot \mathcal{H}_1(\mathsf{md}))M_1$. We replace the individual summands of M_2 one after another by a randomly sampled group elements. Indistinguishability between the adjacent hybrid experiments can be shown via a reduction to the DDH assumption. In the reduction, we receive (A, B, C), use A for the public key corresponding to the replaced summand (if any), the MAC component M_1 by B and the replaced summand by C. Note that we can still issue further MACs in the issuance oracle during the reduction as we can compute the respective term also as vA, $(v \cdot \mathsf{sk}_c)A$ or $(v \cdot \mathcal{H}_1(\mathsf{md}))A$. Finally, we note that the reduction is still capable of verifying received MACs, even without knowing all secret MAC key components, as we have changed the MAC verification to only accept MACs that have been computed as a linear combination of previously issued MACs on the same message (secret client key and metadata hash). This verification can be conducted in the generic group model without knowledge of any MAC key components.
- In the final experiment, we do not use the challenge private bit ok*. Hence, it is obvious that $\Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},12,0}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\mathsf{MAC},12,1}(\lambda) = 1]$. As there is a polynomial number of hybrids and the change in the output distribution between any two adjacent hybrids is at most negligible,it follows that $|\Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{pbp-pv}}_{\Pi_{\mathsf{MAC}},\mathcal{A},1}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$, which completes the proof.

D Instantiations of the Zero-Knowledge Proofs

This section presents instantiations of the zero-knowledge proof systems utilized by our EQS-based and MAC-based constructions. The presented proof systems are standard generalized Schnorr proofs. We assume $\mathcal{H}$ to be a random oracle receiving arbitrary inputs and returning a random element in $\mathbb{Z}_p$.

The proof system

$$(\mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{dlog}}.\mathsf{Verify})$$

for relation

$$\mathsf{ZKP}_{\mathsf{dlog}} = \mathsf{ZKP}\Big\{x : Y = xG\Big\}$$

is instantiated as follows.

Zero-knowledge proof 1: Discrete logarithm

```
$\mathsf{ZKP_{dlog}}.\mathbf{Prove}(x, (G, Y))$ :
(1) Sample $r_x \xleftarrow{\$} \mathbb{Z}_p$ and compute $U_y \leftarrow r_x G$.
(2) Compute $c \leftarrow \mathcal{H}(G, Y, U_y)$.
(3) Compute $\gamma_x \leftarrow r_x - c \cdot x$.
(4) Output $\pi = (c, \gamma_x)$.
$\mathsf{ZKP_{dlog}}.\mathbf{Verify}(\pi, (G, Y))$ :
(5) Compute $W_y \leftarrow cY + \gamma_x G$.
(6) Compute $c' \leftarrow \mathcal{H}(G, Y, W_y)$.
(7) Return 1 if $c = c'$ and 0 otherwise.
```

The proof system

$$(\mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eq}}.\mathsf{Verify})$$

for relation

$$\mathsf{ZKP}_{\mathsf{eq}} = \mathsf{ZKP}\Big\{x : Y_1 = xG_1 \land Y_2 = xG_2\Big\}$$

is instantiated as follows.

Zero-knowledge proof 2: Discrete logarithm equality

```
$\mathsf{ZKP_{eq}}.\mathbf{Prove}(x, (G_1, G_2, Y_1, Y_2))$ :
(1) Sample $r_x \xleftarrow{\$} \mathbb{Z}_p$ and compute $U_b \leftarrow r_x G_b$ for $b \in [2]$.
(2) Compute $c \leftarrow \mathcal{H}(\{G_b, Y_b, U_b\}_{b \in [2]})$.
(3) Compute $\gamma_x \leftarrow r_x - c \cdot x$.
(4) Output $\pi = (c, \gamma_x)$.
$\mathsf{ZKP_{eq}}.\mathbf{Verify}(\pi, (G_1, G_2, Y_1, Y_2))$ :
(5) Compute $W_b \leftarrow cY_b + \gamma_x G_b$ for $b \in [2]$.
```

```
(6)  Compute $c' \leftarrow \mathcal{H}(\{G_b, Y_b, W_b\}_{b\in[2]})$.
(7)  Return 1 if $c = c'$ and 0 otherwise.
```

The proof system

$$(\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify})$$

for relation

$$\mathsf{ZKP}_{\mathsf{4DL}} = \mathsf{ZKP}\Big\{\{x_i\}_{i\in[4]} : Y_i = x_i G \text{ for } i \in [4]\Big\}$$

is instantiated as follows.

Zero-knowledge proof 3: Batched discrete logarithm

```
$\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}(\{x_i\}_{i\in[4]}, (G, \{Y_i\}_{i\in[4]}))$ :
(1)  Sample $r_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $U_i \leftarrow r_i G$ for $i \in [4]$.
(2)  Compute $c \leftarrow \mathcal{H}(G, \{Y_i, U_i\}_{i\in[4]})$.
(3)  Compute $\gamma_i \leftarrow r_i - c \cdot x_i$ for $i \in [4]$.
(4)  Output $\pi = (c, \{\gamma_i\}_{i\in[4]})$.
$\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify}(\pi, (G, \{Y_i\}_{i\in[4]}))$ :
(5)  Compute $W_i \leftarrow cY_i + \gamma_i G$ for $i \in [4]$.
(6)  Compute $c' \leftarrow \mathcal{H}(G, \{Y_i, W_i\}_{i\in[4]})$.
(7)  Return 1 if $c = c'$ and 0 otherwise.
```

The proof system

$$(\mathsf{ZKP}_{\text{eq-or}}.\mathsf{Prove}, \mathsf{ZKP}_{\text{eq-or}}.\mathsf{Verify})$$

for relation

$$\mathsf{ZKP}_{\text{eq-or}} = \mathsf{ZKP}\Big\{(x, b) : Y_1 = xG_1 \wedge Y_2^b = xG_2^b \text{ for } b \in [2]\Big\}$$

is instantiated as follows.

Zero-knowledge proof 4: Dlog equality disjunction

```
$\mathsf{ZKP}_{\text{eq-or}}.\mathsf{Prove}((x, b), (G_1, Y_1\{G_2^i, Y_2^i\}_{i\in[2]}))$ :
(1)  Define $\bar{b} \leftarrow 3 - b$.
(2)  Sample $\gamma_{\bar{b}}, c_{\bar{b}}, r_b \xleftarrow{\$} \mathbb{Z}_p$.
(3)  Compute $U_1^b \leftarrow r_b G_1$ and $U_2^b \leftarrow r_b G_2^b$.
(4)  Compute $U_1^{\bar{b}} \leftarrow c_{\bar{b}} Y_1 + \gamma_{\bar{b}} G_1$ and $U_2^{\bar{b}} \leftarrow c_{\bar{b}} Y_2^{\bar{b}} + \gamma_{\bar{b}} G_2^{\bar{b}}$.
(5)  Compute $c \leftarrow \mathcal{H}(G_1, Y_1\{G_2^i, Y_2^i, U_1^i, U_2^i\}_{i\in[2]})$.
(6)  Compute $c_b = c - c_{\bar{b}}$ and $\gamma_b \leftarrow r_b - c_b \cdot x$.
(7)  Output $\pi = (\{c_i, \gamma_i\}_{i\in[2]})$.
$\mathsf{ZKP}_{\text{eq-or}}.\mathsf{Verify}(\pi, (G_1, Y_1\{G_2^i, Y_2^i\}_{i\in[2]}))$ :
(8)  Compute $W_1^i \leftarrow c_i Y_1 + \gamma_i G_1$ and for $i \in [2]$.
(9)  Compute $c' \leftarrow \mathcal{H}(\{G_b, Y_b, W_b\}_{b\in[2]})$.
(10) Return 1 if $c = c'$ and 0 otherwise.
```

We assume the EQS construction ES to be instantiated according to [22], i.e., with secret key $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \in (\mathbb{Z}_p^*)^\ell$ and $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ such that $\mathsf{pk}_i = \mathsf{sk}_i G_2$ for $i \in \ell$ and G_2 being a generator of $\mathbb{G}_2$. The key verification $\mathsf{ES.VKey}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk})$, hence, verifies that $\mathsf{pk}_i \neq 0$ and $\mathsf{pk}_i = x_i G_2$ for $i \in \ell$. In this work, we consider messages and, hence, keys composed of $\ell = 4$ elements.

The proof system

$$(\mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{eqs}}.\mathsf{Verify})$$

for relation

$$\begin{aligned}\mathsf{ZKP}_{\mathsf{eqs}} &= \mathsf{ZKP}\Big\{\mathsf{sk} : \mathsf{ES.VKey}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}) = 1\Big\} \\ &= \mathsf{ZKP}\Big\{\{x_i\}_{i\in[4]} : x_i \neq 0 \wedge y_i = x_i G_2\Big\}\end{aligned}$$

can be instantiated equivalently to the proof system ($\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Prove}$, $\mathsf{ZKP}_{\mathsf{4DL}}.\mathsf{Verify}$) with the only difference, that we consider generator G_2 of group $\mathbb{G}_2$ instead of generator G and additional reject proofs if any $\mathsf{pk}_i = 0$ for $i \in [\ell]$. The latter ensures that all x_i are non-zero.

The proof system

$$(\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Prove}, \mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Verify})$$

for relation

$$\mathsf{ZKP}_{\mathsf{PdOr}} = \mathsf{ZKP}\Big\{(x, u, v, b) : C_b = uG + xH \\ \wedge M_1 = vG \wedge M_2 = xM_1 + vK_b \text{ for } b \in [2]\Big\}.$$

is instantiated as follows.

Zero-knowledge proof 5: Pedersen MAC equality disjunction

```
$\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Prove}((x, u, v, b), (G, H, M_1, M_2, \{C_i, K_i\}_{i\in[2]}))$ :
(1)  Define $\bar{b} \leftarrow 3 - b$.
(2)  Sample $\gamma_x^{\bar{b}}, \gamma_u^{\bar{b}}, \gamma_v^{\bar{b}}, r_x^b, r_u^b, r_v^b, c^{\bar{b}} \xleftarrow{\$} \mathbb{Z}_p$.
(3)  Compute $U_C^b \leftarrow r_u^b G + r_x^b H,\ U_{M_1}^b \leftarrow r_v^b G$,
     and $U_{M_2}^b \leftarrow r_x^b M_1 + r_v^b K_b$.
(4)  Compute $U_C^{\bar{b}} \leftarrow c^{\bar{b}} C_{\bar{b}} + \gamma_u^{\bar{b}} G + \gamma_x^{\bar{b}} H,\ U_{M_1}^{\bar{b}} \leftarrow c^{\bar{b}} M_1 + \gamma_v^{\bar{b}} G$,
     and $U_{M_2}^{\bar{b}} \leftarrow c^{\bar{b}} M_2 + \gamma_x^{\bar{b}} M_1 + \gamma_v^{\bar{b}} K_{\bar{b}}$.
(5)  Compute $c \leftarrow \mathcal{H}(G, H, M_1, M_2, \{C_i, K_i, U_C^i, U_{M_1}^i, U_{M_2}^i\}_{i\in[2]})$.
(6)  Compute $c^b = c - c^{\bar{b}}$
     and $\gamma_x^b \leftarrow r_x^b - c^b \cdot x,\ \gamma_u^b \leftarrow r_u^b - c^b \cdot u,\ \gamma_v^b \leftarrow r_v^b - c^b \cdot v$.
(7)  Output $\pi = (\{c^i, \gamma_x^i, \gamma_u^i, \gamma_v^i\}_{i\in[2]})$.
$\mathsf{ZKP}_{\mathsf{PdOr}}.\mathsf{Verify}(\pi, (G, H, M_1, M_2, \{C_i, K_i\}_{i\in[2]}))$ :
(8)  Compute $W_C^i \leftarrow c^i C_i + \gamma_u^i G + \gamma_x^i H,\ W_{M_1}^i \leftarrow c^i M_1 + \gamma_v^i G$,
     and $W_{M_2}^i \leftarrow c^i M_2 + \gamma_x^i M_1 + \gamma_v^i K_i$ for $i \in [2]$.
(9)  Compute $c' \leftarrow \mathcal{H}(G, H, M_1, M_2, \{C_i, K_i, W_C^i, W_{M_1}^i, W_{M_2}^i\}_{i\in[2]})$.
(10) Return 1 if $c = c'$ and 0 otherwise.
```