Anonymous Tokens with Private Metadata Bit

Ben Kreuter[1], Tancrède Lepoint[1], Michele Orrù[2,3,4], and Mariana Raykova[1]

[1] Google, {benkreuter,tancrede,marianar}@google.com
[2] Département d'informatique de l'ENS, ENS, CNRS, PSL University, Paris, France, first.last@ens.fr
[3] INRIA, Paris, France
[4] Recurse Center, New York, USA

January 23, 2020

Abstract. We present a cryptographic construction for anonymous tokens with private metadata bit, a primitive that enables an issuer to provide a user with anonymous trust tokens that can embed a single private metadata bit, which is accessible only to the party who holds the secret authority key and is private with respect to anyone else. Our construction provides unforgeability, unlinkability and privacy for the metadata bit properties.

1 Introduction

We present a new cryptographic construction for the primitive *anonymous tokens with private metadata bit*. An anonymous token functionality enables an authority to issue a trust token to a user when it knows the identity of the user. This token can be used at a later time by the user to prove that she is trusted, however, the token also has an *anonymity* property which guarantees that the token does not reveal any additional information about the identity of the user. Nobody except the issuer who has the secret key should be able to issue tokens. The Privacy Pass construction [DGS+18] provides an anonymous token scheme that provides *unforgeability* and *unlinkability*.

We extend the notion of anonymous token with the concept of a *private metadata bit*, which allows the issuer to convey an additional bit of information with the token, for example, whether this user is whitelisted or blacklisted. This additional bit should remain hidden from the user and should be readable only by the party who has the secret key. While this private metadata bit allows the issuer to distinguish tokens with different private metadata bit values, we require an anonymity property which guarantees that tokens with the same metadata bit remain indistinguishable. The scheme should also remain unforgeable.

The Privacy Pass construction employs an oblivious verifiable PRF evaluation where the PRF used is $F(k,t) = k\mathsf{H}(t)$ (using elliptic-curve additive notation). This construction provides verifiability for the token issuance, which assures the user that the issued token has been generated using a previously committed secret key for the issuer. The verifiability is realized using a Schnorr proof for equality of discrete logarithms (DLEQ). The main idea for the extension with a private metadata bit is to use two committed keys, each of which is used for one of the values for the metadata bit, and to give an OR proof for equality of discrete logarithms (DLEQOR) with respect to the two committed keys.

The above idea does not work directly since the token issuance in Privacy Pass is a deterministic algorithm and this enables an attack where the user can learn whether two tokens have been issued with the same private metadata bit value. To circumvent this issue, we generalize the anonymous token construction to a randomized version using the following function for token computation $F(K := (x,y),t) = x\mathsf{H}(t) + y\mathsf{H}'(s)$ where t is the user input to be signed and s is a random value chosen by the issuer.[5] In order to preserve verifiability we replace the Schnorr style DLEQ proof with an Okamoto-Schnorr generalized version of it. Now we can apply the idea of using two keys to enable a private metadata bit.

[5] Note that at token redemption, the user will not send s but send a randomized value $S = r\mathsf{H}'(s)$; hence the issuer cannot de-anonymize the user using s.

One last wrinkle in the security proof is whether the adversary for the unforgeability and the privacy of the metadata bit properties should have access to a verification oracle for tokens of its choice. This is not explicitly supported in the current Privacy Pass security proof [DGS+18]. We provide a new proof for unforgeability of Privacy Pass in the presence of a verification oracle based on a different hardness assumption, the *Chosen Target Gap Diffie-Hellman* assumption, which is a formalization of the Chosen Target Diffie-Hellman in a Gap DH group, which has been defined and used before [BLS01]. In the context of anonymous tokens with private metadata bit we distinguish a verification oracle which just returns one bit about the validity of the token and a verification functionality which returns the value of the private metadata bit (which could be 0, 1 or invalid, and in some applications, e.g. blacklisting, we can merge the states of value 0 and invalid bit). We present a third anonymous token construction that provides unforgeability and privacy for the metadata bit even when the adversary has verification oracle access for the validity of the token, but we crucially require that the adversary *does not get an oracle access to the verification and reading of the private metadata bit.*

Section 2 contains the security assumptions and primitives used throughout the paper, and Section 3 defines the anonymous token primitive and its security defitinions. Next, Section 4 overviews the construction of Privacy Pass in our framework. We present our randomized extension of the Privacy Pass scheme in Section 5, and how we can enable a private metadata bit in the context of this extension in Section 6. Finally, we combine the previous two constructions in Section 7 to obtain an anonymous token with private metadata bit that supports validity verification oracles in the security games for unforgeability and privacy of the metadata bit.

2 Preliminaries

Notation. When sampling the value x uniformly at random from the set S, we write $x \leftarrow_\$ S$. When sampling the value x from the probabilistic algorithm M, we write $x \leftarrow \mathsf{M}$. We use $:=$ to denote assignment. For an integer $n \in \mathbb{N}$, we denote with $[n]$ the interval $\{0, \ldots, n-1\}$. We denote vectors in bold. For a vector $\mathbf{a}$, we denote with a_i the i-th element of $\mathbf{a}$.

The output resulting form the interaction of two (interactive) algorithms $\mathsf{A}, \mathsf{B} \in \mathrm{PPT}$ is denoted as $[\![a, b]\!] \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$. If only the first party receives a value at the end of the interaction, we write $a \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$ instead of $[\![a, \bot]\!] \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$. For a probabilistic algorithm M running on input x, we denote with $[\mathsf{M}(x)]$ the *range* of outputs, i.e., all possible outputs of M on x occurring with non-zero probability.

We assume the existence of a group generator algorithm $\mathsf{GrGen}(1^\lambda)$ that, given as input the security parameter in unary form outputs the description $\Gamma = (\mathbb{G}, p, G, H)$ of a group $\mathbb{G}$ of prime order p; G and H are two nothing-up-my-sleeve (NUMS) generators of $\mathbb{G}$. For simplicity, we will assume that the prime p is of length λ.

2.1 Security assumptions

We describe here the assumptions we will use throughout the paper.

Discrete Logarithm. The discrete logarithm assumption for a group generator GrGen states that given a tuple of elements (G, H) where $G \leftarrow_\$ \mathbb{G}$ and $H \leftarrow_\$ \mathbb{G}$, any PPT adversary has negligible advantage in returning $h \in \mathbb{Z}_p$ such that $H = hG$. That is,

$$\mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 1.

Decisional Diffie-Hellman. The decisional Diffie-Hellman (DDH) assumption for a group generator GrGen states that given a tuple of elements $(P, A := aP, B := bP)$ where $P \leftarrow_\$ \mathbb{G}$, and $a, b \leftarrow_\$ \mathbb{Z}_p$, any adversary $\mathsf{A} \in \mathrm{PPT}$ has negligible advantage in distinguishing $C \leftarrow_\$ \mathbb{G}$ from the Diffie-Hellman $C = abP$. That is,

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) := \left|\Pr\left[\mathrm{DDH}^0_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right] - \Pr\left[\mathrm{DDH}^1_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where $\mathrm{DDH}^\beta_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 1.

Game $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$	Game $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$	Game $\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$
$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$\Gamma, := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$X \leftarrow\!\!\$\, \mathbb{G}$	$P \leftarrow\!\!\$\, \mathbb{G}$	$P \leftarrow\!\!\$\, \mathbb{G}$
$x \leftarrow \mathsf{A}(\Gamma, X)$	$a \leftarrow\!\!\$\, \mathbb{Z}_p; \quad A := aP$	$a \leftarrow\!\!\$\, \mathbb{Z}_p; \quad A := aP$
return $(xG = X)$	$b \leftarrow\!\!\$\, \mathbb{Z}_p; \quad B := bP$	$b \leftarrow\!\!\$\, \mathbb{Z}_p; \quad B := bP$
	$C_0 := abP; \quad C_1 \leftarrow\!\!\$\, \mathbb{G}$	$C \leftarrow \mathsf{A}(\Gamma, P, A, B)$
	$b' \leftarrow \mathsf{A}(\Gamma, P, A, B, C_\beta)$	**return** $(C = abP)$
	return b'	

Fig. 1. The games for discrete logarithm, decisional Diffie-Hellman, and computational Diffie-Hellman.

Game $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$	Oracle $\textsc{Target}(t_i)$	Oracle $\textsc{Help}(Y)$
$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	**if** $\exists (t_i, Y_i)$ to T	$q := q + 1$
$x \leftarrow\!\!\$\, \mathbb{Z}_p$	**return** Y_i	**return** xY
$X := xG$	**else**	
$q := 0; \quad T := [\,]$	$Y_i \leftarrow\!\!\$\, \mathbb{G}$	
$\{t_i, Z_i\}_{i=0}^{\ell-1} \leftarrow \mathsf{A}^{\textsc{Target},\textsc{Help}}(\Gamma, X)$	append (t_i, Y_i) to T	
return $(t_i, Y_i) \in T$ all different **and** $xY_i = Z_i \;\; \forall i \in [\ell]$ **and** $q < \ell$	**return** Y_i	

Fig. 2. The Chosen-Target Diffie-Hellman security game.

Computational Diffie-Hellman. The computational Diffie-Hellman (CDH) assumption for a group generator GrGen states that given a tuple of elements $(P, A := aP, B := bP)$ where $P \leftarrow\!\!\$\, \mathbb{G}$, and $a, b \leftarrow\!\!\$\, \mathbb{Z}_p$, any adversary $\mathsf{A} \in \mathrm{PPT}$ has negligible probability in outputting $C = abP$. That is,

$$\mathsf{Adv}^{\mathrm{cdh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

where $\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 1.

Chosen-target Diffie-Hellman. The chosen-target Diffie-Hellman (CTDH) assumption [Bol03, HL06] for the group generator GrGen states that all $\mathsf{A} \in \mathrm{PPT}$ have negligible advantage in solving CDH on $\ell + 1$ target group elements, even if the adversary is given access to a CDH helper oracle for ℓ instances. More formally:

$$\mathsf{Adv}^{\mathrm{ctdh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 2. Note that for $q = 0$ and $|T| = 1$, the game is equivalent to CDH.

The Chosen-target Diffie-Hellman (CTDH) assumption [Bol03], where an adversary is able to observe some honest "handshakes", has proven itself very useful for a range of applications: Password-Authenticated Key Exchanges [JKX18], signatures [LQ04], and private-set intersection [JL10]. One can also find it called in the literature as *One-More Diffie-Hellman* (1MDH) assumption [BP02, BNPS03], both in the "chosen-target" flavor (where the adversary can chose the $\ell+1$ subset of challenges to solve) and the "known-target" flavor (where the adversary must solve a fixed set of $\ell+1$ challenges). In the known-target version of one-more Diffie-Hellman, an adversary receives as input a group description Γ, a group element $X = xG \in \mathbb{G}$, and challenges $(Y_0, \ldots, Y_\ell) \in \mathbb{G}^{\ell+1}$ sampled uniformly at random. The adversary wins the game $\mathrm{1MDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ if it outputs $Z_i = xY_i$ for all $i \in [\ell + 1]$, even when given access to at CDH oracle for at most ℓ arbitrary elements. Known-target one-more Diffie-Hellman is equivalent to chosen-target Diffie-Hellman [KM08].

Game $\mathrm{KSND}_{\Pi,\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda)$	Game $\mathrm{ZK}^{\beta}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)$	Oracle $\textsc{Prove}_{\beta}(\phi, w)$
$\Gamma \coloneqq \mathsf{S}(1^\lambda); (\sigma, \tau) \leftarrow \Pi.\mathsf{S}(\Gamma)$	$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$	**if** $(\phi, w) \notin \mathsf{R}$ **then return** $\bot$
$r \leftarrow\!\!\$\ \{0,1\}^{\mathsf{A.rl}(\lambda)}; (\phi, \pi) \coloneqq \mathsf{A}(\sigma; r)$	$(\sigma, \tau) \leftarrow \Pi.\mathsf{S}(\Gamma)$	$\pi_0 \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w)$
$w \leftarrow \mathsf{Ext}(\sigma, r)$	$b' \leftarrow \mathsf{A}^{\textsc{Prove}_{\beta}}(\sigma)$	$\pi_1 \leftarrow \Pi.\mathsf{Sim}(\sigma, \tau, \phi)$
return $(\Pi.\mathsf{V}(\sigma, \phi, \pi)$ **and** $\mathsf{R}(\phi, w) =$ **false**$)$	**return** b'	**return** π_β

Fig. 3. Games for knowledge soundness (KSND), and zero knowledge (ZK).

2.2 Non-interactive arguments of knowledge

A non-interactive zero-knowledge (NIZK) argument of knowledge Π for a relation R consists of the following three algorithms:

- $(\sigma, \tau) \leftarrow \Pi.\mathsf{S}(\Gamma)$, the common reference string (CRS) generation algorithm that outputs a CRS σ together with some trapdoor information τ.
- $\pi \leftarrow \Pi.\mathsf{P}(\sigma, \phi, w)$, a prover which takes as input some $(\phi, w) \in \mathsf{R}$ and a CRS σ, and outputs a proof π.
- $bool \leftarrow \Pi.\mathsf{V}(\sigma, \phi, \pi)$ a verifier that, given as input a statement ϕ together with a proof π outputs **true** or **false**, indicating acceptance of the proof.

To ease notation for prover and verifier, we will assume that the group description Γ can be inferred from the CRS σ. Π must satisfy three basic properties: completeness, knowledge soundness, and zero-knowledge A NIZK is complete if every correctly generated proof verifies. More formally, Π is (perfectly) *complete* if for any $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, $\sigma \in [\Pi.\mathsf{S}(\Gamma)]$ and $(\phi, w) \in \mathsf{R}$:

$$\Pr[\Pi.\mathsf{V}(\sigma, \phi, \Pi.\mathsf{P}(\sigma, \phi, w))] = 1.$$

Knowledge soundness [BG93] means that for any prover able to produce a valid proof there exists an efficient algorithm, which has access to the prover's random coins, capable of extracting a witness for the given statement. More precisely, Π is *knowledge-sound* for R if for any PPT adversary A there exists a PPT extractor Ext such that:

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi,\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda) \coloneqq \Pr\left[\mathrm{KSND}_{\Pi,\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda)\right] = \mathsf{negl}(\lambda)$$

and $\mathrm{KSND}_{\Pi,\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda)$ is defined in Fig. 3 and $\mathsf{A.rl}(\lambda)$ is the randomness length for the machine A. An *argument of knowledge* is a knowledge-sound proof system.

A proof system Π for R is *zero-knowledge* if no information about the witness is leaked by the proof. Pragmatically, this is shown by specifying an additional PPT algorithm $\Pi.\mathsf{Sim}$, that takes as input the trapdoor information τ and a statement ϕ, and outputs a valid proof π indistinguishable from those generated via $\Pi.\mathsf{P}$. Formally, Π is zero-knowledge if for any PPT adversary A:

$$\mathsf{Adv}^{\mathrm{zk}}_{\Pi,\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{ZK}^{0}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)\right] - \Pr\left[\mathrm{ZK}^{1}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)\right]\right| = \mathsf{negl}(\lambda)\,.$$

where $\mathrm{ZK}^{\beta}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)$ is defined in Fig. 3.

Note that, in our constructions, we will use Sigma proof protocol made non-interactive using the Fiat-Shamir transform in the random oracle model. More specifically we will be using proof systems for discrete logarithms equality (DLEQ) defined by the following language

$$\mathcal{L}_{DLEQ,\Gamma} \coloneqq \left\{ (X, T, S, W) \in \mathbb{G}^4 \ : \ \exists (x, y) \in \mathbb{Z}_p^2 \ , \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\},$$

and OR of discrete logarithms equalities (DLEQOR) defined by the following language

$$\mathcal{L}_{DLEQOR,\Gamma} \coloneqq \left\{ (\mathbf{X} \coloneqq (X_0, X_1), T, S, W) \in \mathbb{G}^5 \ : \ \aleph(b, x, y) \in [2] \times \mathbb{Z}_p^2 \ . \ \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}.$$

We provide these ZK protocol in Appexdix B.

3 Anonymous Tokens

In this section we describe building block primitives that we will use in our constructions, the functionality of the anonymous tokens primitive as well as the security properties that we will aim to achieve.

3.1 Anonymous token functionality

We describe two flavors of the anonymous token functionality: the basic anonymous token functionality enables a user to obtain a token from an issuer that authenticates a particular value t that the user has. The issuer has a public commitment of the key it uses to authenticate the value and the user can verify that the resulting token was correctly issued and will verify correctly with the corresponding private key. The validity of each token can be verified using the secret issuance key.

The second variant is an anonymous token with a private metadata bit. In this functionality the issuer has an additional input bit during the token issuance and it is used to tag the token. The user can verify the validity of the token with respect the committed parameters but she cannot learn what the private bit is (we formalize this security property later).

The following definition captures both functionalities where the shaded text refers only to the anonymous token with private metadata bit.

Definition 1 (Anonymous Token). *An* anonymous token with private metadata bit *scheme* AT *consists of the following algorithms:*

- $(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$ – *a key generation algorithm that generates a private key* sk *and a commitment to the private key* $\mathsf{Com}_{\mathsf{sk}}$ *(note this is* not *a public verification key)*
- $\mathsf{token} \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, b) \rangle$ – *a signing protocol that involves the interactive algorithms* $\mathsf{AT.Usr}$ *(run by the user), and* $\mathsf{AT.Sig}$ *(run by the issuing server). At the end of the interaction, the user receives either* $\mathsf{token} \coloneqq \perp$ *or* $\mathsf{token} \coloneqq (t, \sigma)$ *where* σ *is a signature corresponding to her input value* $t \in \{0,1\}^\lambda$*, and the server which has as input its private key and a bit* b *does not obtain any output. We refer to the pair* (t, σ) *as an anonymous token.*
- $bool \leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$ – *a verification algorithm that takes as input a token* (t, σ) *and a private key. It returns a boolean indicating the validity of the token.*
- $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, (t, \sigma))$ – *an algorithm that takes as input a token* (t, σ) *and a private key. It returns an indicator value* $\mathsf{ind} \in \{\perp, 0, 1\}$ *which either returns the value of the private metadata bit or the invalid bit.*

An anonymous token protocol AT *is* correct*, if any honestly-generated token verifies for validity and the correct private metadata bit is retrieved , i.e. for all* $(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \in [\mathsf{AT.KeyGen}(1^\lambda)]$*, all* $t \in \{0,1\}^*$*, and* $b \in \{0,1\}$ *:*

$$\mathsf{Pr}[\mathsf{AT.VerValid}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, b) \rangle) = 1] = 1,$$
$$\mathsf{Pr}[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, b) \rangle) = b] = 1,$$
$$\mathsf{Pr}[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, b) \rangle) = b' \mid b' \neq b] < \mathsf{negl}(\lambda).$$

We proceed to define the security properties relevant for the anonymous tokens.

3.2 Unforgeability

The first security property that we want from an anonymous token is *unforgeability*, which guarantees that no party that does not have the secret key can generate valid anonymous tokens. In particular an adversary who obtains ℓ valid tokens cannot generate $\ell+1$ valid tokens. And in the case when there is private metadata bit value, the adversary can query a token oracle ℓ times for each bit value, but should not be able to generate $\ell + 1$ valid tokens that have the same private metadata bit value.

Definition 2 (One-More Unforgeability). *An anonymous token scheme* AT *is* one-more unforgeable*, if for all* $\mathsf{A} \in PPT$*, all* ℓ, n, n'*:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell,n,n'}(\lambda) \coloneqq \Pr\left[\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell,n,n'}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell,n,n'}(\lambda)$ *is defined as follows*

Game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell,n,n'}(\lambda)$

$(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$

$\mathcal{O}(\cdot, \cdot) \coloneqq \langle \cdot, \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, \cdot)\rangle$

$(t_i, \sigma_i)_{i=1}^{\ell+1} \leftarrow \mathsf{A}^{\mathcal{O}(\cdot,\cdot),\ \mathsf{AT.VerValid}(\mathsf{sk},\cdot),\ \mathsf{AT.ReadBit}(\mathsf{sk},\cdot)}(\mathsf{Com}_{\mathsf{sk}})$

return t_i *all different* ***and*** $\mathsf{AT.VerValid}(\mathsf{sk}, (t_i, \sigma_i)) = 1\ \forall i \in [\ell + 1]$

and *there exists a bit* b' *such that* $\mathsf{AT.ReadBit}(\mathsf{sk}, (t_i, \sigma_i)) = b'$.

where A *can invoke the signing protocol* $\langle \cdot, \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, \cdot)\rangle$ ℓ *times for each private metadata bit value, the validity verification oracle* $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot)$ n *times, and the bit reading oracle* $\mathsf{AT.ReadBit}(\mathsf{sk}, \cdot)$ n' *times.*

3.3 Unlinkability

The next security property is concerned with the user anonymity and guarantees that an issuer cannot link a token to a particular execution of the signing protocol. More if the user and the server have executed the signing protocol m times and later the issuer is given a valid token, we limit the probability that it can guess for which execution this token was coming.

Definition 3 (Unlinkability). *An anonymous token scheme* AT *is* κ-unlinkable*, if for all adversaries* $\mathsf{A} \in PPT$ *and for all* $m \in \mathbb{Z}, m > 0$*:*

$$\mathsf{Adv}^{\mathrm{unlink}}_{\mathsf{AT},\mathsf{A},m}(\lambda) \coloneqq \Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] \leq \frac{\kappa}{m} + \mathsf{negl}(\lambda)\,,$$

where $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)$ *is defined as follows:*

Game $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)$

$(\mathsf{st}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{A}(1^\lambda)$

$\forall \ell \in [m], t_\ell \leftarrow_{\$} \{0,1\}^\lambda$

$\forall \ell \in [m], (\mathsf{st}, (t_\ell, \sigma_\ell)) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t_\ell), \mathsf{A}(\mathsf{st}, \ell)\rangle$

$j \leftarrow_{\$} [1, m]$

$j' \leftarrow \mathsf{A}(\mathsf{st}, (t_j, \sigma_j))$

return $(j' = j)$ ***and*** $\forall \ell \in [m], (t_\ell, \sigma_\ell) \neq \perp$

where st *denotes the state of the adversary.*

3.4 Private metadata bit

The last security property that we define concerns only anonymous tokens with private metadata bit. It guarantees that a user cannot learn any information about the private metadata bit associated with the token she receives. Intuitively, our definition guarantees that an adversary who can obtain tokens for messages of its choice with metadata bit of its choice, an arbitrary number of tokens for messages with the fixed challenge bit, and can access a validity verification oracle for the tokens, cannot guess the challenge bit with a probability non-negligibly better than 1/2.

Definition 4 (Private Metadata Bit). *An anonymous token scheme* AT *provides* private metadata bit *if for all adversary* $\mathsf{A} \in PPT$ *the advantage:*

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{AT},\mathsf{A}}(\lambda) := \left|\Pr\left[\mathrm{PMB}^{0}_{\mathsf{AT},\mathsf{A}}(\lambda)\right] - \Pr\left[\mathrm{PMB}^{1}_{\mathsf{AT},\mathsf{A}}(\lambda)\right]\right| < \mathsf{negl}(\lambda)\,,$$

where $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$ *is defined as follows:*

Game $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$

$(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^{\lambda})$

$\mathcal{O}(\cdot,\cdot) := \langle \cdot, \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, \cdot)\rangle$

$\beta' \leftarrow \mathsf{A}^{\mathcal{O}(\cdot,\cdot),\mathcal{O}(\cdot,\beta),\mathsf{AT.VerValid}(\mathsf{sk},\cdot)}(\mathsf{Com}_{\mathsf{sk}})$

return β'

and the adversary A *has access to the following oracles: a signing oracle* $\langle \cdot, \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, \cdot)\rangle$ *where* A *can provide both the message and the bit to be used for the token, a signing oracle* $\langle \cdot, \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, \beta)\rangle$ *with the fixed challenge bit* $\beta \in \{0,1\}$ *where the adversary can provide only the message, and a validity verification oracle* $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot)$ *where* A *provide a token for verification.*

4 Privacy Pass

We start by recalling, using our notation, the anonymous token protocol proposed in [DGS+18] (under the name Privacy Pass) and built on top of the VOPRF as described in [JKK14] (under the name 2Hash). Note that this anonymous token protocol is deterministic, i.e., there will exist a unique value $\sigma \in \mathbb{G}$ corresponding to a string $\{0,1\}^{\lambda}$ such that (t, σ) is a valid token. This property will make difficult to directly extend the construction to support private metadata bit. In the following sections, we will generalize Privacy Pass that enables randomized tokens (Section 5) and will eventually extend the construction to support private metadata bit (Sections 6 and 7).

The Privacy Pass construction uses a Schnorr-style DLEQ proof for the verifiability in the issuance phase [DGS+18]. Its proof of unforgeability (without validity oracle) was originally based on the One-More-Decryption security of El Gamal. We will show in Theorem 9 that it can be proved under the chosen-target Diffie–Hellman assumption, and prove in Appendix A that the assumptions are equivalent.

Construction 1 *Let* $\Gamma := (\mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^{\lambda})$ *be an algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs a random generator* G*. Let* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a proof system for the DLEQ relationship defining the language*

$$\mathcal{L} := \left\{ (X,T,W) \in \mathbb{G}^3 \;:\; \nexists x \in \mathbb{Z}_p \;,\; \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} \right\}.$$

We construct an anonymous token scheme AT *defined by the following algorithms:*

- $(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^{\lambda})$*:*
  - *Run* $\Gamma := (\mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^{\lambda})$ *to obtain group parameters.* Γ *will be an implicit input to all other algorithms.*

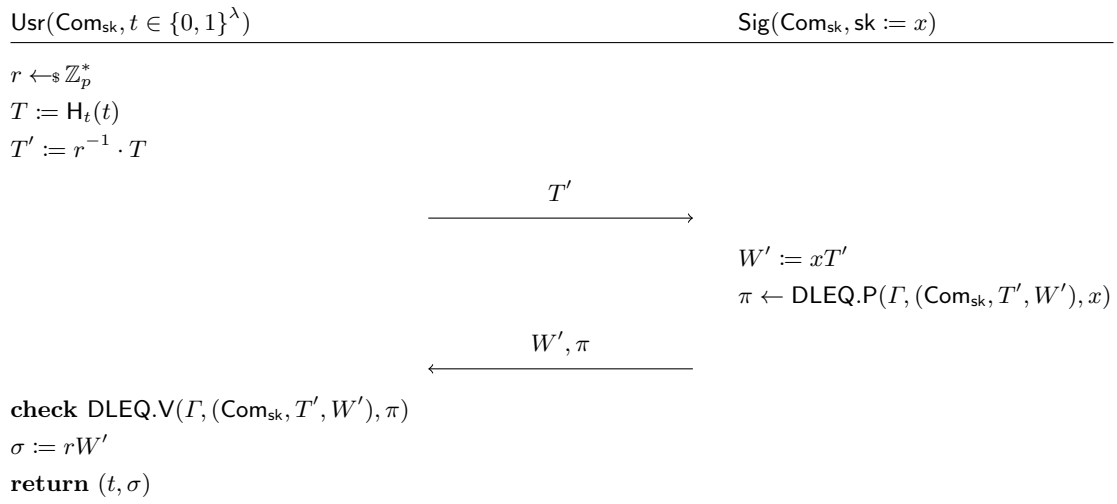

Fig. 4. Privacy Pass Issuance Protocol.

- *Sample a random invertible value* $x \leftarrow\!\!\$\ \mathbb{Z}_p$*, and set* $\mathsf{sk} := x$*,* $\mathsf{Com}_{\mathsf{sk}} := xG$*.*

– $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Figure 4.*

– $bool \leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$*: if* $\sigma = x\mathsf{H}_t(t)$*, return* 1*. Otherwise, return* 0*.*

Correctness. The protocol never aborts: this follows by perfect correctness of the underlying proof system DLEQ. Therefore, the client always returns a tuple $(t, \sigma) \in \{0,1\}^\lambda \times \mathbb{G}^2$ such that

$$\sigma = rW' = r(xT') = xT = x\mathsf{H}_t(t).$$

Security. We willl prove unforgeability and unlinkeability of Privacy Pass in Theorems 9 and 11.

5 Okamoto-Schnorr Privacy Pass

In this section we describe a novel anonymous token protocol that builds on top of Privacy Pass [DGS+18] (Section 4). Our anonymous token protocol can be viewed as a generalization of Privacy Pass that enables *randomized* tokens, which will be an important property when we extend the construction to support private metadata bit (Sections 6 and 7). While the Privacy Pass construction uses a Schnorr-style DLEQ proof for the verifiability in the issuance phase, we use the corresponding Okamoto-Schnorr-style [Oka93] variant of the DLEQ proof protocol. A different approach towards randomization of the deterministic evaluation algorithm could be leveraging pairings in the spirit of the construction for partially oblivious PRFs [ECS+15] while maintaining capabilities for token verification and unlinkability from the issuer. We do not pursue this approach here for efficiency reasons.

Construction 2 *Let* $\Gamma := (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be an algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs two distinct random generators* G *and* H*. Let* $\mathsf{H}_s \colon \mathbb{G} \times \{0,1\}^* \to \mathbb{G}$ *be a random oracle mapping a group element and a string into group elements. Let* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a proof system for the DLEQ relationship defining the language*

$$\mathcal{L}_{DLEQ,\Gamma} := \left\{ (X, T, S, W) \in \mathbb{G}^4 \ : \ \exists (x, y) \in \mathbb{Z}_p^2 \ , \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}.$$

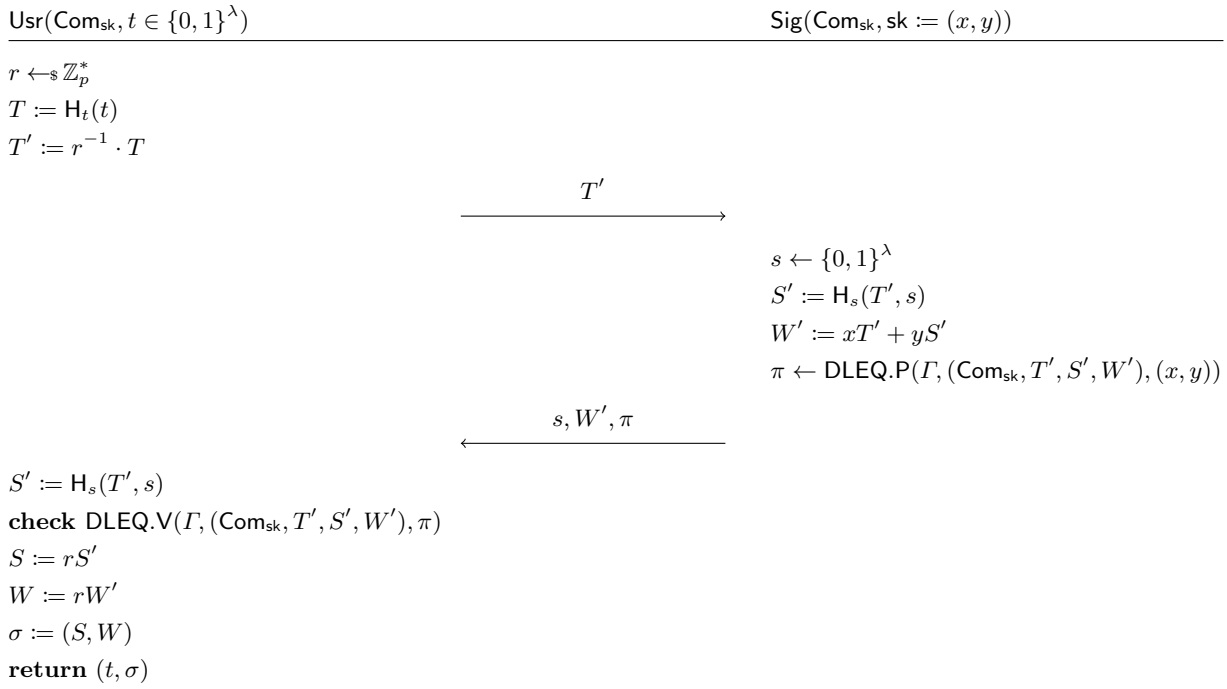

Fig. 5. Okamoto–Schnorr Privacy Pass Issuance Protocol.

We construct an anonymous token scheme AT *defined by the following algorithms:*

- $(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run* $\Gamma \coloneqq (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *to obtain group parameters.* Γ *will be an implicit input to all other algorithms.*
  - *Sample a pair of random invertible values* $(x, y) \leftarrow_\$ \mathbb{Z}_p^2$*, and set* $\mathsf{sk} \coloneqq (x, y)$*,* $\mathsf{Com}_{\mathsf{sk}} \coloneqq xG + yH$*.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Figure 5.*
- *bool* $\leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$*:*
  - *Parse* $\sigma = (S, W)$*.*
  - *If* $W = x\mathsf{H}_t(t) + yS$*, return* 1*. Otherwise, return* 0*.*

Remark 5. A observation of the above protocol is that if we set $y = 0$, then we obtain the Privacy Pass protocol defined in Section 4.

Correctness. The protocol never aborts: this follows by perfect correctness of the underlying proof system DLEQ. Therefore, the client always returns a tuple $(t, (S, W)) \in \{0,1\}^\lambda \times \mathbb{G}^2$ such that

$$W = rW' = r(xT' + yS') = xT + xS = x\mathsf{H}_t(t) + yS.$$

5.1 Unforgeability

We show that our construction is one-more unforgeable. In particular, this will show that a client cannot use knowledge of already signed tokens, received during the issuance phase, to produce more valid tokens.

Oracle $\textsc{Sign}(\mathsf{Com_{sk}}, \mathsf{sk}, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$.

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$S' \coloneqq \mathsf{H}(T', s)$
$W' \coloneqq xT' + yS'$
$\pi \leftarrow \mathsf{DLEQ.P}(\Gamma, \mathsf{Com_{sk}}, T', S', W', x, y)$
$\pi \leftarrow \mathsf{DLEQ.Sim}(\Gamma, \mathsf{Com_{sk}}, T', S', W')$
return (s, W', π)

Fig. 6. Summary of hybrid changes for unforgeability. Changes with respect to the previous hybrid are highlighted in light gray .

One-More unforgeability without validity oracle. We start by first proving that our construction provides one-more unforgeability when the adversary does *not* have access to the AT.VerValid oracle.

Lemma 6. *Let $\Gamma \coloneqq (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be a parameter generation function such that H is a random element in the group (i.e., the discrete log of H with respect to G is unknown),* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a DLEQ zero-knowledge proof system. Construction 2 is one-more unforgeable assuming the hardness of the chosen-target Diffie–Hellman (CTDH) problem when the adversary does not have access to the* AT.VerValid *oracle.*

Proof. We denote by $\textsc{Sign}(\mathsf{Com_{sk}}, \mathsf{sk}, T')$ the signing oracle that performs the issuer part of the signing protocol (Fig. 6) and prove this lemma using a hybrid argument. First, we replace the proving algorithm by the zero-knowledge simulator DLEQ.Sim, and then show a direct reduction to the chosen-target Diffie–Helman problem.

Hyb_1 This is the game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell,n}(\lambda)$: the adversary is provided with the commitment $\mathsf{Com_{sk}}$. The adversary has access to the signing oracle and the various random oracles H_s (used for the response of the server), H_t (used for blinding the message by the user), and H_c (used for computing the DLEQ proof). At the end of its execution, it outputs $\ell + 1$ tokens.

Hyb_2 This hybrid replaces the way zero-knowledge proofs are generated when answering signing oracles: instead of using the proving algorithm DLEQ.P, we use the zero-knowledge simulator DLEQ.Sim for all signing queries.
If there exists an adversary $\mathsf{A} \in \mathrm{PPT}$ whose advantage is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of DLEQ. Consider the adversary $\mathsf{B} \in \mathrm{PPT}$ for the game $\mathrm{ZK}^\beta_{\mathsf{DLEQ}}(\lambda)$. B generates correctly the commitment $\mathsf{Com_{sk}}$, and generate the proofs π's in the signing queries via the $\textsc{Prove}_b$ oracle for the statement $(\mathsf{Com_{sk}}, T', S', W') \in \mathcal{L}_{DLEQ,\Gamma}$. At the end of the execution, A returns β. If A wins the game, then B outputs 1, otherwise it outputs 0. We have

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQ},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) \right| .$$

We will prove that if there is an adversary A that has non-negligible advantage $\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda)$, then we can construct an adversary B that has non-negligible advantage in the chosen-target Diffie–Hellman game $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$.

Assuming the existence of A, we construct B as follows. B receives the group description and $A \in \mathbb{G}$ as input, samples $y \leftarrow \mathbb{Z}_p$ invertible, and computes $\mathsf{Com_{sk}} \coloneqq A + yH$. Then, it runs $\mathsf{A}(\mathsf{Com_{sk}})$. Note that $\mathsf{Com_{sk}}$ is distributed as in $\mathsf{Hyb}_{\mathsf{Hyb}_2}$. We need now to specify how B answers oracle queries. The adversary B overrides the queries to the random oracles H_t and $\textsc{Sign}$ in the following way:

– to any query to the oracle $\mathsf{H}_t(t)$, the adversary B invokes the oracle $\textsc{Target}(t)$ and returns whatever it returns;

– to any query of the form $\textsc{Sign}(T')$, the adversary B select $s \leftarrow\!\!\$\, \{0,1\}^\lambda$ and defines $S' \coloneqq \mathsf{H}_s(T', s)$. Then it invokes the oracle $Z \coloneqq \textsc{Help}(T')$, defines $W' = Z + yS'$, and simulates the proof π. Finally, it returns (s, W', π).

All other random oracle queries are left unchanged. First, note that the distributions of H_t and $\textsc{Sign}$ are identical to the ones of $\mathsf{Hyb}_{\mathsf{Hyb}_2}$. At the end of the execution, A returns $\ell+1$ tuples $(t_i, (S_i, W_i)) \in \{0,1\}^\lambda \times \mathbb{G}^2$, and B finally returns $\{t_i, W_i - yS_i\}_{i=0}^{\ell}$.

We claim that the adversary B wins the game $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ every time that A wins. By the winning condition of game $\mathsf{Hyb}_{\mathsf{Hyb}_2}$, A won if and only if all t_i are different and $W_i = x\textsc{Target}(t_i) + yS_i$ where x is the unique element of $\mathbb{Z}_p$ such that $A = xG$. Furthermore, by the winning condition, A only called the signing oracle $\textsc{Sign}$ at most $q_S \leq \ell$ times; therefore $\textsc{Help}$ was called at most ℓ times.

Finally, this shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell,n}(\lambda) \coloneqq \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda).$$

and this concludes the proof. □

Handling a validity oracle. The previous proof only handles the case where an adversary does not have access to a validity oracle (more precisely, to the $\mathsf{AT.VerValid}$ oracle). However, in practice, when a user is redeeming anonymous tokens, it is expected that the behavior will change depending on whether this token was valid or not. In other words, when deploying anonymous tokens, an adversary is likely to be able to learn, by submitting $(t, (S, W))$ as token, whether $(X - yH, \mathsf{H}_t(t), W - yS)$ is a valid Diffie–Hellman tuple, for t, S, W of its choice. Note that this holds both for our construction and for Privacy Pass: giving access to the $\mathsf{AT.VerValid}$ oracle enables the adversary to learn if $(X, \mathsf{H}_t(t), W)$ is a DDH tuple for inputs t, W of its choice.

This specific behavior is not unique to our anonymous tokens definition. In particular, a similar issue arised many times in the literature, including (without being exhaustive):

– when proving the CCA security of the (hashed) El Gamal encryption scheme [ABR01, CKS09];
– when proving the unforgeability of Chaum's undeniable signature scheme [CA89, Cha90, OP01];
– blind signatures [BLS01];
– the VOPRF [JKK14] on which Privacy Pass is based,
– and so on.

Instead, all these schemes are proved under a *gap* problem [OP01], a computational problem that gives oracle access to the underlying decision problem.[6] In particular, [OP01] defines the Gap DH problem, which given a triple (P, aP, bP), ask to find the element abP with the help of a Decision Diffie–Hellman oracle (which answers whether (X, Y, Z) is a valid DH triple).

Interestingly, the chosen-target DH problem was originally introduced by Boldyreva [Bol03] in *Gap DH* groups [BLS01], that is groups where CDH is assumed to be hard but DDH is assumed to be easy. In other words, the original definition of CTDH was proposed in groups where the adversary has access to a DDH oracle that reveals if a tuple is a valid DDH tuple, while our definition of CTDH in Section 2.1 did not required the group $\mathbb{G}$ to be a Gap DH group. We therefore formalize the notion of *chosen-target gap Diffie-Hellman* (CTGDH) problem, the gap problem equivalent to the CTDH problem.

Definition 7 (Chosen-target gap Diffie-Hellman.). *The chosen-target gap Diffie-Hellman assumption for the group generator* GrGen *states that all* $\mathsf{A} \in PPT$ *have negligible advantage in solving CDH on* $\ell + 1$ *target group elements, even if the adversary is given access to a DDH oracle, and to a CDH oracle for* ℓ *instances. More formally:*

$$\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) \coloneqq \Pr\left[\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ *is defined in Fig. 7.*

[6] We follow the name usage of [OP01, BLS01, JKK14], but the Gap DH problem is also known under the name of *strong* Diffie-Hellman problem [ABR01, CKS09].

Game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$	Oracle $\textsc{Target}(t_i)$	Oracle $\textsc{Help}(Y)$
$\Gamma \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	**if** $\exists (t_i, Y_i)$ to T	$q \coloneqq q + 1$
$x \leftarrow\!\!\$\, \mathbb{Z}_p$; $X \coloneqq xG$	**return** Y_i	**return** xY
$q \coloneqq 0$; $T = [\,]$	**else**	
$\{t_i, Z_i\}_{i=0}^{\ell-1} \leftarrow \mathsf{A}^{\textsc{Target},\textsc{Help},\textsc{Ddh}}(\Gamma, X)$	$Y_i \leftarrow\!\!\$\, \mathbb{G}^*$	Oracle $\textsc{Ddh}(t, W)$
return $(t_i, Y_i) \in T$ all different **and** $xY_i = Z_i \;\; \forall i \in [\ell]$	append (t_i, Y_i) to T	**return** $W \stackrel{?}{=} x\textsc{Target}(t)$
and $q < \ell$	**return** Y_i	

Fig. 7. The Chosen-Target Gap Diffie-Hellman assumption.

One-More unforgeability with validity oracle. We now prove that our construction provides one-more unforgeability under the chosen-target gap Diffie–Hellman assumption.

Theorem 8. *Let $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be a parameter generation function such that H is a random element in the group (i.e., the discrete log of H with respect to G is unknown), $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ be a DLEQ zero-knowledge proof system. Construction 2 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

Proof. The only difference with the proof of Lemma 6 is that we need to specify how the queries to $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot, \cdot)$ are answered to. For any query to $\mathsf{AT.VerValid}(\mathsf{sk}, t_i, \sigma_i)$, the challenger (who knows the secret y) will (1) parse $\sigma_i = (S_i, W_i)$, (2) let $X_i = W_i - yS_i$, and (3) return whatever $\textsc{Ddh}(t_i, X_i)$ returns. Thus, we can handle any number of verification queries. This shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell,n}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda).$$

and concludes the proof. □

Application to Privacy Pass. As mentioned above, the one-more unforgeability security notion from Privacy Pass [DGS+18] did not provide the adversary with a validity oracle. One-more unforgeability without validity oracle is proven under the one-more decryption of El Gamal problem, which we prove to be equivalent to the chosen-target Diffie–Hellman problem in Appendix A. We therefore prove the following theorem:

Theorem 9. *Let $(\mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ be a parameter generation function, $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ be a DLEQ zero-knowledge proof system. Construction 1 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

Proof. The proof follows directly from the proofs of Lemma 6 and Theorem 8. We detail here the differences:

- In the hybrid, the proof system is with respect to the language $\mathcal{L}$ instead of $\mathcal{L}_{DLEQ,\Gamma}$.
- In the reduction to chosen-target gap Diffie-Hellman, B sets $\mathsf{Com}_{\mathsf{sk}} \coloneqq A$, and answers the signing queries with whatever $\textsc{Help}$ answers and the simulated proof. At the end of the game, it outputs whatever A outputs.

This shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell,n}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda).$$

and concludes the proof. □

5.2 Unlinkability

In this section, we will prove that Construction 2 is 1-unlinkeable (cf. Definition 3), which means that the probability that an adversary can guess which of m issued tokens is redeemed is upper-bounded by $1/m + \mathsf{negl}(\lambda)$.

Theorem 10. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a DLEQ zero-knowledge proof system. Construction 2 provides* 1*-unlinkability according to Definition 3, assuming the hardness of DDH.*

$\mathsf{Hyb}_1(\lambda)$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $t_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda, r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^*,$
  $T_\ell \coloneqq \mathsf{H}_t(t_\ell), T'_\ell \coloneqq r_\ell^{-1} T_\ell;$
$(\mathsf{st}, \mathsf{Com_{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\mathsf{DLEQ.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  $i \leftarrow\!\!\$\ [m]$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S'$
  $W \coloneqq r_i W'_i$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_2(\lambda)$ $\boxed{\mathsf{Hyb}_3(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $t_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda, r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^*,$
  $T_\ell \coloneqq \mathsf{H}_t(t_\ell), T'_\ell \coloneqq r_\ell^{-1} T_\ell;$
$(\mathsf{st}, \mathsf{Com_{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\mathsf{DLEQ.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQ.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
  $i \leftarrow\!\!\$\ [m];$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S'$
  $W \coloneqq r_i W'_i;$ $\boxed{W = x_i T_i + y_i S}$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_4(\lambda)$ $\boxed{\mathsf{Hyb}_5(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $T'_\ell \leftarrow\!\!\$\ \mathbb{G}^*$
  $r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^*$
  $t_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda$
  $\mathsf{H}(t_\ell) \coloneqq T_\ell \coloneqq r_\ell T'_\ell;$
$(\mathsf{st}, \mathsf{Com_{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\exists k \in [m], \mathsf{DLEQ.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQ.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
  $i \leftarrow\!\!\$\ [m];$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S';$ $\boxed{S \leftarrow\!\!\$\ \mathbb{G}}$
  $W = x_i T_i + y_i S$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_6(\lambda)$ $\boxed{\mathsf{Hyb}_7(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $T'_\ell \leftarrow\!\!\$\ \mathbb{G}^*;$
  $r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^*$
  $t_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda$
  $\mathsf{H}(t_\ell) \coloneqq T_\ell \coloneqq r_\ell T'_\ell;$
$(\mathsf{st}, \mathsf{Com_{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\mathsf{DLEQ.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQ.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
  $i \leftarrow\!\!\$\ [m];$
  $j \leftarrow\!\!\$\ [m]$
  if not$(x, y) \coloneqq (x_i, y_i) \coloneqq (x_j, y_j)$: **abort**
  $S' \coloneqq S'_i$
  $S \leftarrow\!\!\$\ \mathbb{G}$
  $W = xT_i + yS;$ $\boxed{W = xT_j + yS}$
  $(t, \sigma) \coloneqq (t_j, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

Fig. 8. Summary of hybrid changes for unlinkability.

Proof. First, let us introduce some notation. Since the signing protocol is one round, for $\ell \in [1, m]$, we will unroll the step

$$(\mathsf{st}, \mathsf{token}_\ell) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t_\ell), \mathsf{A}(\mathsf{st}, \ell) \rangle$$

as

$$\begin{aligned}
&r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^* \\
&T_\ell \coloneqq \mathsf{H}_t(t_\ell) \\
&T'_\ell \coloneqq r_\ell^{-1} T_\ell \\
&(\mathsf{st}, (s_\ell, W'_\ell, \pi_\ell)) \leftarrow \mathsf{A}(\mathsf{st}, \ell, T'_\ell) \\
&S'_\ell \coloneqq \mathsf{H}_s(T'_\ell, s_\ell) \\
&\textbf{check} \quad \mathsf{DLEQ.V}(\Gamma, (\mathsf{Com}_{\mathsf{sk}}, T'_\ell, S'_\ell, W'_\ell), \pi_\ell) \\
&S_\ell \coloneqq r_\ell S'_\ell \\
&W_\ell \coloneqq r_\ell W'_\ell \\
&\sigma_\ell \coloneqq (S_\ell, W_\ell) \\
&\mathsf{token}_\ell \coloneqq (t_\ell, \sigma_\ell)
\end{aligned} \tag{1}$$

The first step of our proof is to show that the adversary's view in the unlinkability game when it is provided challenge token i is indistinguishable from an execution where it is presented with any other challenge token j. We present the sequence of hybrids in Figure 8 which allows us to transitions between these two executions with negligible distinguishing probability.

Hyb_1 The first hybrid is the execution of the unlinkability game where the adversary is given as a challenge the i-th token, and we have explicitly unrolled the interaction between the issuer and the challenger.

Hyb_2 We now use the soundness extractor for the DLEQ proof to extract the witnesses for all issued tokens and we abort if the the extracted witnesses are not correct. This hybrid is indistinguishable from the previous by the knowledge soundness of the proof system. Hence we have

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQ},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|.$$

Hyb_3 At this point, instead of computing W by unblinding the element W' provided by the adversary, we compute it ourselves using the extracted witnesses as $W \coloneqq xT_i + yS$. If, by contradiction, there exists the outcome of the experiments in Hyb_3 and Hyb_3 is different, it must be the case that $W = xT_i + yS \neq rW'$. That is, $W' \neq xT'_i + yS'$. However, this would mean that (x, y) is not a valid witness for the statement (X, T'_i, S', W'), which contradicts the relationship check introduced in Hyb_6.

Hyb_4 In the next hybrid, we change how we compute the values T'_ℓ. Instead of computing them from T_ℓ, we sample them at random and program accordingly the RO at the values t_ℓ. The distribution of this hybrid is identical to the previous one.

Hyb_5 The experiment proceeds exactly as in the previous game, except now S is sampled uniformly at random from $\mathbb{G}$. The rest of the game proceeds as in the previous hybrid.
If it exists $\mathsf{A} \in \mathrm{PPT}$ for which the outcome of the two hybrids is different, then it is possible to construct an adversary $\mathsf{B} \in \mathrm{PPT}$ for $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ by exploiting the random self-reducibility property of DDH.
The adversary B takes as input the group description together with a tuple $(P, A \coloneqq aP, B \coloneqq bP, C) \in \mathbb{G}^4$ where $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$ and has to distinguish $C \coloneqq abP$ (the case $\beta = 0$) from a uniformly distributed element over $\mathbb{G}$ (the case $\beta = 1$). B runs the game as per Hyb_3, except instead of sampling T'_ℓ uniformly at random from $\mathbb{G}$, B sets them as $T'_\ell = \gamma_\ell P$ where γ_ℓ is random for all $\ell \in [m]$. Instead of computing $T_\ell = r_\ell T'_\ell$, B sets $T_\ell = \gamma_\ell \alpha_\ell A + \gamma_\ell \alpha'_\ell P = (\alpha_\ell a + \alpha'_\ell)\gamma_\ell P = (\alpha_\ell a + \alpha'_\ell)T'_\ell$, i.e. implicitly setting $r_\ell \coloneqq \alpha_\ell a + \alpha'_\ell$. For every query $\mathsf{H}_s(T'_\ell, s)$, B samples $\beta_{\ell,s}, \beta'_{\ell,s}$ at random and programs $\mathsf{H}_s(T'_\ell, s) = \gamma_\ell \beta_{\ell,s} B + \gamma_\ell \beta'_{\ell,s} P = (\beta_{\ell,s} b + \beta'_{\ell,s})\gamma_\ell P$. Finally, B sets $S \coloneqq \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell C + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell A + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell B + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P$. If C is random, then S is random

the distributions coincides with Hybrid Hyb_6. If $C = abP$, then

$$\begin{aligned} S &= \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell C + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell A + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell B + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P \\ &= \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell (abP) + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell (aP) + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell (bP) + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P \\ &= \alpha_\ell \beta_{\ell,s_\ell} ab (\gamma_\ell P) + \alpha_\ell a \beta'_{\ell,s_\ell} (\gamma_\ell P) + \alpha'_\ell b \beta_{\ell,s_\ell} (\gamma_\ell P) + \alpha'_\ell \beta'_{\ell,s_\ell} (\gamma_\ell P) \\ &= (\alpha_\ell a + \alpha'_\ell)(\beta_{\ell,s_\ell} b + \beta'_{\ell,s_\ell})(\gamma_\ell P)\,, \end{aligned}$$

which results in the distribution of Hybrid Hyb_5. It follows therefore that the advantage in distinguishing the two hybrids is, for all $\mathsf{A} \in \mathrm{PPT}$:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

Hyb_6 In this hybrid, we sample two indices $i, j \in [m]$, and if the witnesses are different the game aborts. We note that, if the game aborts at this step, by soundness of the DLEQ proof system we would have that $\mathsf{Com}_{\mathsf{sk}} = x_i G + y_i H = x_j G + y_j H$ and thus $(y_j - y_i)/(x_i - x_j)$ is the discrete log of H base G. (if the two witnesses are different, it must be that $x_i \neq x_j$, and thus the inverse of $(x_i - x_j)$ exists.) It is therefore possible to construct an adversary $\mathsf{B} \in \mathrm{PPT}$ for $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$: the adversary B obtains a group description $\Gamma := (p, \mathbb{G}, \tilde{G})$ and a challenge $\tilde{H}$. It sets $G := \tilde{G}$ and $H := \tilde{H}$ and runs exactly as per Hyb_6. If the two proofs verify, it extracts the two witnesses (x_i, y_i), and (x_j, y_j) and returns $(y_j - y_i)/(x_i - x_j)$. The adversary wins every time that Hyb_6 aborts.

Hyb_7 At this point, we swap the computation of W and use T_j instead of T_i. We remark that, in fact, r_i and r_j are only used in order to compute T_i and T_j. Therefore, the inputs to the adversary T'_i, T'_j perfectly hide T_i and T_j. Therefore, the tokens t_i and t_j can be swapped when programming the random oracle, and this hybrid is perfectly indistinguishable from Hyb_5.

We obtain that

$$\mathsf{Adv}^{\mathsf{Hyb}_8}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + \mathsf{negl}(\lambda)\,.$$

Now it is sufficient to bound the probability of success adversary in Hyb_8. Since the index that the adversary needs to guess there is independent of the challenge it receives we conclude that

$$\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] \leq \frac{1}{m} + \mathsf{negl}(\lambda)\,.$$

which completes the proof. □

Application to Privacy Pass. In [DGS+18], the Privacy Pass protocol is proved to be unconditionally unlinkable (up to the soundness error of the proof system). This can be recovered from the proof above by removing in the proof all the elements about the elements s, S', y, H and S. In particular, all hybrids become indistinguishable assuming the soundness of the proof system, which yields the following theorem.

Theorem 11. *Let* $(\mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function,* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a DLEQ zero-knowledge proof system. Construction 1 provides* 1*-unlinkability according to Definition 3.*

6 Anonymous Token With Private Metadata Bit

In this section, we present an extension of the anonymous token construction from Section 5 that supports a private metadata bit, as specified in Definition 1. The high level idea is that we will use different secret keys for tokens that have different private metadata bit values and will commit to both secret keys. In order to hide which bit is associated with the token, we will use DLEQOR proofs instead of DLEQ proofs, which will convince the user that one of the two commitments has been used.

Remark 12. Our construction will *not* provide a (meaningful) AT.VerValid functionality, but only a AT.ReadBit functionality. The reason for this is that this scheme cannot provide privacy for the metadata bit if the user can access to a validity oracle, which is given to the adversary in the security Definition 4. We will use this construction as a building block for a scheme in Section 7 that will provide a validity verification.

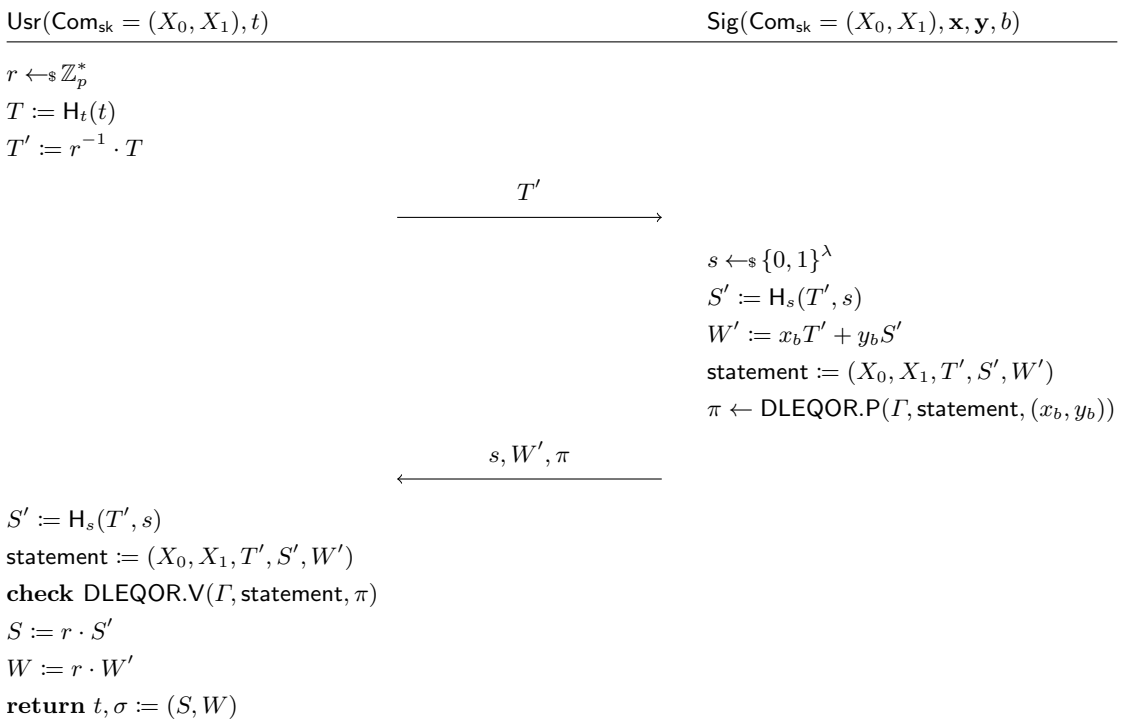

Fig. 9. Token Issuance for Tokens with Private Metadata Bit.

Construction 3 *Let* $\Gamma \coloneqq (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be an algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs two distinct random generators* G *and* H*. Let* $\mathsf{H}_s \colon \mathbb{G} \times \{0,1\}^* \to \mathbb{G}$ *be a random oracle mapping a group element and a string into group elements. Let* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a proof system for the DLEQOR relationship defining the language*

$$\mathcal{L}_{DLEQOR,\Gamma} \coloneqq \left\{ (X_0, X_1, T, S, W) \in \mathbb{G}^5 \ : \ \exists (b, x, y) \in \{0,1\} \times \mathbb{Z}_p^2 \ . \ \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}.$$

We construct an anonymous token scheme AT *defined by the following algorithms:*

- $(\mathsf{Com}_s k, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run* $\Gamma \coloneqq (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *to obtain group parameters.* Γ *will be an implicit input to all other algorithms.*
  - *Sample a pair of random values* $(\mathbf{x} \coloneqq (x_0, x_1), \mathbf{y} \coloneqq (y_0, y_1)) \leftarrow\!\!\$\ \mathbb{Z}_p^2$*, and set* $\mathsf{sk} \coloneqq (\mathbf{x}, \mathbf{y})$*, and* $\mathsf{Com}_{\mathsf{sk}} \coloneqq \mathbf{X} = (x_0 G + y_0 H, x_1 G + y_1 H)$*.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Figure 9.*

– $bool \leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$: *return* 1.
– $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, (t, \sigma))$:
  • *Parse* $\sigma = (S, W)$.
  • *If* $W = x_0\mathsf{H}_t(t) + y_0 S$ *return* 0, *else if* $W = x_1\mathsf{H}_t(t) + y_1 S$ *return* 1, *else return* $\perp$.

Correctness. The protocol never aborts: this follows by perfect correctness of the underlying proof system DLEQOR. Therefore, the client always returns a tuple $(t, (S, W)) \in \{0,1\}^\lambda \times \mathbb{G}^2$ such that there exists $b \in \{0,1\}$ such that

$$W = rW' = r(x_b T' + y_b S') = x_b T + x_b S = x_b \mathsf{H}_t(t) + y_b S.$$

If there exists $i, i' \in [n]$ such that $i \neq i'$ and $W = x_i \mathsf{H}_t(t) + y_i S = x_{i'} \mathsf{H}_t(t) + y_{i'} S$ it means that:

$$\mathsf{H}_t(t) = \frac{x_i - x_{i'}}{y_{i'} - y_i} S.$$

However, the left hand side of the equation is distributed uniformly at random from $\mathbb{G}$ and independently from the terms on the right-hand side. The terms of the right hand-side are distributed uniformly at random as well. Therefore, the probability that more than one term satisfies the verification equation is the probability that two elements sampled uniformly at random over $\mathbb{G}$ are equal. This probability is $1/p$. It follows that

$$\Pr[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{pk}), \mathsf{AT.Sig}(\mathsf{sk}, b)\rangle) = b] = 1,$$
$$\Pr[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{pk}), \mathsf{AT.Sig}(\mathsf{sk}, b)\rangle) = b' \mid b' \neq b] = 1 - \frac{1}{p} < \mathsf{negl}(\lambda).$$

6.1 Unforgeability

Oracle $\textsc{Sign}(T', b')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$.

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$S' \coloneqq \mathsf{H}(T', s)$
$W' \coloneqq xT' + yS'$
$\pi \leftarrow \mathsf{DLEQOR.P}(\Gamma, \mathsf{Com}_{\mathsf{sk}}, T', S', W', x_{b'}, y_{b'})$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, (\mathsf{Com}_{\mathsf{sk}}, T', S', W'))$
return (s, W', π)

Fig. 10. Summary of hybrid changes for unforgeability. Changes with respect to the previous hybrid are highlighted in light gray .

Theorem 13. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 3 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

Proof. We denote by $\textsc{Sign}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}, T')$ the signing oracle that performs the issuer part of the signing protocol (Fig. 10) and prove this lemma using a hybrid argument. First, we replace the proving algorithm of our construction AT (Construction 3) by the zero-knowledge simulator DLEQOR.Sim, and then show that an adversary A that can win the security game for one-more unforgeability with non-negligible probability, can be used to construct an adversary B who can win the security game for one-more unforgeability for AT′ (Construction 2).

Hyb_1 This is the game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell,n,n'}(\lambda)$: the adversary is provided with the commitment $\mathsf{Com}_{\mathsf{sk}}$. The adversary has access to the signing oracle and the various random oracles H_s (used for the response of the server), H_t (user for bliding the message by the user), and H_c (used for computing the DLEQ proof). At the end of its execution, it outputs $\ell+1$ tokens for the same bit b'.

Hyb_2 This hybrid replaces the way zero-knowledge proofs are generated when answering signing oracles: instead of using the proving algorithm DLEQOR.P, we use the zero-knowledge simulator DLEQOR.Sim for all signing queries.

If there exists an adversary $\mathsf{A} \in \mathrm{PPT}$ whose advantage is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of DLEQOR. Consider the adversary $\mathsf{B} \in \mathrm{PPT}$ for the game $\mathrm{ZK}^{\beta}_{\mathsf{DLEQOR}}(\lambda)$ that generates correctly the commitment $\mathsf{Com}_{\mathsf{sk}}$. B generates the proofs π's in the signing queries via the $\textsc{Prove}_\beta$ oracle for the statement $(\mathsf{Com}_{\mathsf{sk}}, T', S', W') \in \mathcal{L}_{DLEQOR,\Gamma}$. At the end of the execution, A returns β. If A wins the game, then B outputs 1, otherwise it outputs 0. We have

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQOR},\mathsf{B}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

Now, assume there exists an adversary A that can win the security game for one-more unforgeability with non-negligible probability p. We construct an adversary B who can win the security game for one-more unforgeability for AT' (Construction 2) with probability $p/2$ as follows.

B first draws a bit $b \in \{0,1\}$ which will correspond to its guess on which bit A will succeeds for the one-more unforgeability game. Upon reception of $X_b = x_b G + y_b H$ for an unknown tuple (x_b, y_b) in the CTGDH game, it samples x_{1-b}, y_{1-b} in $\mathbb{Z}_p$, sets $X_{1-b} \coloneqq x_{1-b}G + y_{1-b}H$, and defines $\mathsf{Com}_{\mathsf{sk}} \coloneqq (X_0, X_1)$. B then sends $\mathsf{Com}_{\mathsf{sk}}$ to A.

We need now to specify how B answers A's queries.

- For any query to $\mathsf{AT}.\mathsf{H}_t$, $\mathsf{AT}.\mathsf{H}_s$, $\mathsf{AT}.\mathsf{H}_c$, it forwards whatever the oracles $\mathsf{AT}'.\mathsf{H}_t$, $\mathsf{AT}'.\mathsf{H}_s$, $\mathsf{AT}'.\mathsf{H}_c$ answer.
- For any query to $\mathsf{AT}.\mathsf{VerValid}$, it answers 1.
- For any query $\mathsf{AT}.\mathsf{Sig}(\mathsf{sk}, T', b')$, it proceeds as follows.
  - If $b' = 1-b$, it correctly follows the issuance protocol described in Fig. 9, i.e., it samples $s \leftarrow\!\!\$\ \{0,1\}^\lambda$, define $S' \coloneqq \mathsf{H}_s(T', s)$, compute $W' = x_{1-b}T' + y_{1-b}S'$, but simulates the DLEQOR proof π for the statement $(\mathsf{Com}_{\mathsf{sk}}, T', S', W') \in \mathcal{L}_{DLEQOR,\Gamma}$. It then answers (s, W', π) to A.
  - If $b' = b$, it invokes $\mathsf{AT}'.\mathsf{Sig}(\mathsf{sk}, T')$ and parse the output as (s, W', π'). It discards π' and simulates the DLEQOR proof π for the statement $(\mathsf{Com}_{\mathsf{sk}}, T', S', W') \in \mathcal{L}_{DLEQOR,\Gamma}$. It then answers (s, W', π) to A.
- For any query $\mathsf{AT}.\mathsf{ReadBit}(\mathsf{sk}, t, (S, W))$, it proceeds as follows.
  - If $W = x_{1-b}\mathsf{H}_t(t) + y_{1-b}S$, it outputs $1-b$.
  - Otherwise, it invoves $\mathsf{AT}'.\mathsf{VerValid}(\mathsf{sk}, t, (S, W))$. If the oracle answers 1 (valid), it answers b; otherwise it answers $\perp$.

Note that the answers to $\mathsf{AT}.\mathsf{Sig}$ and $\mathsf{AT}.\mathsf{ReadBit}$ follow the same distribution as in Hyb_2 because B uses the AT' oracles to sign and verify tokens for the bit it does not know the witness for; additionally, when reading the bit, if the $\mathsf{AT}'.\mathsf{VerValid}$ oracle answers 1, it means that $(t, (S, W))$ is a valid token for the bit b, hence B should answer b.

Finally, B forwards A's answer to the CTGDH challenger: if B guessed correctly the bit on which A produced the forgery, the answer will make it win the CTGDH game. □

6.2 Unlinkability

In this section, we will prove that Construction 3 is **2**-unlinkeable instead of 1-unlinkeable (cf. Definition 3), which means that the probability that an adversary can guess which of m issued tokens is redeemed is upper-bounded by $2/m + \mathsf{negl}(\lambda)$. Indeed, they key idea is that the adversary can embed different private metadata bits during the issuances, halving its search space at most.

Theorem 14. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 3 provides* 2*-unlinkability according to Definition 3 assuming the hardness of DDH.*

Proof. The first step of our proof is to show that the adversary's view in the unlinkability game when it is provided challenge token i is indistinguishable from an execution where it is presented with a challenge token j which has been issued with the same value for the private metadata bit. We present the sequence of hybrids in Figure 11 which allows us to transitions between these two executions with negligible distinguishing probability.

Hyb_1 The first hybrid is the execution of the unlinkability game where the adversary is given as a challenge the i-th token, and we have explicitly unrolled the interaction between the issuer and the challenger.

Hyb_2 In the second hybrid we use the extractor for the DLEQOR proofs to extract the witnesses for all issued tokens and we abort if the the extracted witnesses are not correct. This hybrid is indistinguishable from the previous by the knowledge soundness of the proof system. Hence we have

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQOR},\mathsf{B}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda)\right| .$$

Hyb_3 In the third hybrid, we change the way we sample the challenge index by first sampling one of the two sets of tokens that have the same bit in the witnesses, and then sample the index in that set. The distribution of i is identical as before, therefore the two hybrids are indistinguishable.

Hyb_4 In this hybrid, we compute W directly using the extracted witness. Since $\mathsf{R}((b', x_i, y_i), (\mathsf{Com}_s k, T'_i, S'_i, W'_i))$ is true, Then

$$W'_i = x_i T'_i + y_i S'_i ,$$

and hence

$$W = r_i W'_i = x_i r_i T'_i + y_i r_i S'_i = x_i T_i + y_i S .$$

The two hybrids are therefore indistinguishable.

Hyb_5 In the next hybrid, we change how we compute the values T'_ℓ. Instead of computing them from T_ℓ, we sample them at random and program accordingly the RO at the values t_ℓ. The distribution of this hybrid is identical to the previous one.

Hyb_6 In this hybrid, we change the way we compute the value S used for the computation of the challenge token. We reduce the indistinguishability from the previous hybrid to the hardness of the DDH problem. Assuming that there is an adversary A that distinguishes these two hybrids, we construct an adversary B that distinguishes the DDH challenge $(P, A := aP, B := bP, C) \in \mathbb{G}^4$ where $a, b \leftarrow\!\!\$\, \mathbb{Z}_p$.
Instead of sampling T'_ℓ uniformly at random from $\mathbb{G}$, B sets them as $T'_\ell = \gamma_\ell P$ where γ_ℓ is random for all $\ell \in [m]$. Instead of computing $T_\ell = r_\ell T'_\ell$, B sets $T_\ell = \gamma_\ell \alpha_\ell A + \gamma_\ell \alpha'_\ell P = (\alpha_\ell a + \alpha'_\ell)\gamma_\ell P = (\alpha_\ell a + \alpha'_\ell)T'_\ell$, i.e. implicitly setting $r_\ell := \alpha_\ell a + \alpha'_\ell$. For every query $\mathsf{H}_s(T'_\ell, s)$, B samples $\beta_{\ell,s}, \beta'_{\ell,s}$ at random and programs $\mathsf{H}_s(T'_\ell, s) = \gamma_\ell \beta_{\ell,s} B + \gamma_\ell \beta'_{\ell,s} P = (\beta_{\ell,s} b + \beta'_{\ell,s})\gamma_\ell P$. Finally, B sets $S := \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell C + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell A + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell B + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P$. If C is random, then S is random the distributions coincides with Hybrid Hyb_6. If $C = abP$, then

$$\begin{aligned} S &= \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell C + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell A + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell B + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P \\ &= \alpha_\ell \beta_{\ell,s_\ell} \gamma_\ell (abP) + \alpha_\ell \beta'_{\ell,s_\ell} \gamma_\ell (aP) + \alpha'_\ell \beta_{\ell,s_\ell} \gamma_\ell (bP) + \alpha'_\ell \beta'_{\ell,s_\ell} \gamma_\ell P \\ &= \alpha_\ell \beta_{\ell,s_\ell} ab (\gamma_\ell P) + \alpha_\ell a \beta'_{\ell,s_\ell} (\gamma_\ell P) + \alpha'_\ell b \beta_{\ell,s_\ell} (\gamma_\ell P) + \alpha'_\ell \beta'_{\ell,s_\ell} (\gamma_\ell P) \\ &= (\alpha_\ell a + \alpha'_\ell)(\beta_{\ell,s_\ell} b + \beta'_{\ell,s_\ell})(\gamma_\ell P) , \end{aligned}$$

which results in the distribution of Hybrid Hyb_5. It follows therefore that the advantage in distinguishing the two hybrids is, for all $\mathsf{A} \in$ PPT:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{B}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

$\mathsf{Hyb}_1(\lambda)$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $t_\ell \leftarrow_\$ \{0,1\}^\lambda, r_\ell \leftarrow_\$ \mathbb{Z}_p^*,$
  $T_\ell \coloneqq \mathsf{H}_t(t_\ell), T'_\ell \coloneqq r_\ell^{-1} T_\ell;$
$(\mathsf{st}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\exists k \in [m], \mathsf{DLEQOR.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  $i \leftarrow_\$ [m]$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S'$
  $W \coloneqq r_i W'_i$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_2(\lambda)$ $\mathsf{Hyb}_3(\lambda)$, $\boxed{\mathsf{Hyb}_4(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $t_\ell \leftarrow_\$ \{0,1\}^\lambda, r_\ell \leftarrow_\$ \mathbb{Z}_p^*,$
  $T_\ell \coloneqq \mathsf{H}_t(t_\ell), T'_\ell \coloneqq r_\ell^{-1} T_\ell;$
$(\mathsf{st}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\exists k \in [m], \mathsf{DLEQOR.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQOR.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
    for $\mathsf{ind} = 0, 1$
      $J_{\mathsf{ind}} \coloneqq \{\ell \in [m] \mid b_\ell = \mathsf{ind}\}$
  $i \leftarrow_\$ [m]$; $b' \leftarrow \mathsf{Ber}(|J_1|/m),\ i \leftarrow_\$ J_{b'}$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S'$
  $W \coloneqq r_i W'_i$; $\boxed{W = x_i T_i + y_i S}$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_5(\lambda)$ $\boxed{\mathsf{Hyb}_6(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $T'_\ell \leftarrow_\$ \mathbb{G}^*$
  $r_\ell \leftarrow_\$ \mathbb{Z}_p^*$
  $t_\ell \leftarrow_\$ \{0,1\}^\lambda$
  $\mathsf{H}(t_\ell) \coloneqq T_\ell \coloneqq r_\ell T'_\ell;$
$(\mathsf{st}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\exists k \in [m], \mathsf{DLEQOR.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQOR.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
    for $\mathsf{ind} = 0, 1$
      $J_{\mathsf{ind}} \coloneqq \{\ell \in [m] \mid b_\ell = \mathsf{ind}\}$
  $b' \leftarrow \mathsf{Ber}(|J_1|/m),\ i \leftarrow_\$ J_{b'}$
  $S' \coloneqq S'_i$
  $S \coloneqq r_i S';$ $\boxed{S \leftarrow_\$ \mathbb{G}}$
  $W = x_i T_i + y_i S$
  $(t, \sigma) \coloneqq (t_i, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

$\mathsf{Hyb}_7(\lambda)$ $\boxed{\mathsf{Hyb}_8(\lambda)}$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
for $\ell \in [m]$
  $T'_\ell \leftarrow_\$ \mathbb{G}^*;$
  $r_\ell \leftarrow_\$ \mathbb{Z}_p^*$
  $t_\ell \leftarrow_\$ \{0,1\}^\lambda$
  $\mathsf{H}(t_\ell) \coloneqq T_\ell \coloneqq r_\ell T'_\ell;$
$(\mathsf{st}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{A}(1^\lambda)$
$(\mathsf{st}, \{(s_\ell, W'_\ell, \pi_\ell)\}_{\ell\in[m]}) \leftarrow \mathsf{A}(\mathsf{st}, \{T'_\ell\}_{\ell\in[m]})$
$\forall \ell \in [m], S'_\ell \coloneqq \mathsf{H}_s(T_\ell, s_\ell);$
if $\exists k \in [m], \mathsf{DLEQOR.V}(\Gamma, X, T'_k, S'_k, W'_k, \pi_k) = 0$
  $\sigma \coloneqq \perp$
else :
  for $\ell \in [m]$
    $(b_\ell, x_\ell, y_\ell) \leftarrow \mathsf{DLEQOR.Ext}(\Gamma, X, T'_\ell, S'_\ell, W'_\ell, \pi_\ell)$
    if not $\mathsf{R}((b_\ell, x_\ell, y_\ell), (\Gamma, X, T'_\ell, S'_\ell, W'_\ell))$: **abort**
    for $\mathsf{ind} = 0, 1$
      $J_{\mathsf{ind}} \coloneqq \{\ell \in [m] \mid b_\ell = \mathsf{ind}\}$
  $b' \leftarrow \mathsf{Ber}(|J_1|/m),\ i \leftarrow_\$ J_{b'}$
  $j \leftarrow_\$ J_{b'}$
  if not $(x, y) \coloneqq (x_i, y_i) \coloneqq (x_j, y_j)$: **abort**
  $S' \coloneqq S'_i$
  $S \leftarrow_\$ \mathbb{G}$
  $W = xT_i + yS;$ $\boxed{W = xT_j + yS}$
  $(t, \sigma) \coloneqq (t_j, (S, W))$
$i' \leftarrow \mathsf{A}(\mathsf{st}, \sigma)$
return $(i' == i)$

Fig. 11. Summary of hybrid changes for PMB unlinkability.

Hyb_7 In this hybrid, we sample two indices i, j in the same set $J_{b'}$. By definition of $J_{b'}$ and the fact that the witnesses and the statement satisfy the relation, for $\mathsf{Com}_{\mathsf{sk}} = (X_0, X_1)$, it holds that

$$X_{b'} = x_i G + y_i H = x_j G + y_j H\,.$$

If the witnesses are different and the game aborts, then we can solve the discrete logarithm relation between G and H, as in **??** of the proof of Theorem 10.

Hyb_8 At this point, since r_i and r_j are only used in order to compute T_i and T_j, the inputs to the adversary T_i', T_j' perfectly hide T_i and T_j. Therefore, we can swap (t_i, T_i) and (t_j, T_j) in the computation of W and the token. This hybrid is perfectly indistinguishable from the previous one.

We obtain that

$$\mathsf{Adv}^{\mathsf{Hyb}_8}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + \mathsf{negl}(\lambda)\,.$$

Therefore, it is sufficient to bound the advantage of the adversary in Hyb_8, which we do as follows:

$$\begin{aligned}&\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] = \\ &\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1 \mid b' = 0\right]\Pr[b' = 0] + \Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1 \mid b' = 1\right]\Pr[b' = 1]\,.\end{aligned}$$

Assume both J_0 and J_1 are not empty. We have that $\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1 \mid b' = 0\right] = 1/|J_0| + \mathsf{negl}(\lambda)$ since the challenge token is independent of the index i. Similarly $\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1 \mid b' = 1\right] = 1/|J_1| + \mathsf{negl}(\lambda)$. Hence,

$$\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] = \frac{1}{|J_0|}\frac{|J_0|}{m} + \frac{1}{|J_1|}\frac{|J_1|}{m} + \mathsf{negl}(\lambda) = \frac{2}{m} + \mathsf{negl}(\lambda)\,.$$

If either J_0 or J_1 are empty, then

$$\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] \leq \frac{1}{m}\frac{m}{m} + \mathsf{negl}(\lambda) \leq \frac{2}{m} + \mathsf{negl}(\lambda)\,.$$

which completes the proof. □

6.3 Privacy of the Metadata Bit

Theorem 15. *Let $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be a parameter generation function such that H is a random element in the group (i.e., the discrete log of H with respect to G is unknown),* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 3 provides privacy for the metadata bit according to Definition 4 assuming the hardness of DDH.*

Proof. We consider the sequence of hybrids presented in Fig. 12 that transitions from an execution of $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$ to an execution of $\mathrm{PMB}^{1-\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$ (see Definition 4). We argue that each pair of consecutive hybrids are indistinguishable for the adversary and thus $\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{AT},\mathsf{A}}(\lambda) = \mathsf{negl}(\lambda)$. We do not explicitly write the verification validity oracle in the games since in this construction this functionality is dummy and can always be simulated.

Hyb_0 This is the game $\mathrm{PMB}^0_{\mathsf{AT},\mathsf{A}}(\lambda)$: here, the adversary is provided the commitment $\mathsf{Com}_{\mathsf{sk}} := (X_0, X_1)$. The adversary has access to the signing oracle for a bit of its choosing, and a challenge oracle that signs new tokens with the bit b. Additionally, it has access to the random oracles: $\mathsf{H}_t, \mathsf{H}_s, \mathsf{H}_c$. At the end of its execution, it outputs a bit b'.

Hyb_1 In this hybrid, we unroll the procedures for the signing protocol: we generate s, S', W', and π within the security experiment itself. The two games are perfectly indistinguishable.

Oracle $\textsc{Sign}(\hat{b}, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\pi \leftarrow \mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, T', S', W', x_{\hat{b}}, y_{\hat{b}})$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
return (s, W', π)

Oracle $\textsc{Sign}_b(T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$\pi \leftarrow \mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, T', S', W', x_{\hat{b}}, y_{\hat{b}})$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
return (s, W', π)

Oracle $\textsc{Sign}(\hat{b}, T')$ in $\mathsf{Hyb}_4(\lambda), \mathsf{Hyb}_5(\lambda), \mathsf{Hyb}_6(\lambda)$

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
return (s, W', π)

Oracle $\textsc{Sign}_b(T')$ in $\mathsf{Hyb}_4(\lambda)$, $\mathsf{Hyb}_5(\lambda)$, $\boxed{\mathsf{Hyb}_6(\lambda)}$

$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$W' := x_b T' + y' S'$
$W' := x_{1-b} T' + y' S'$
$\boxed{W' := x_{1-b} T' + y_{1-b} S'}$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
return (s, W', π)

Fig. 12. Summary of the proof for private metadata bit, with highlighted changes to the signing oracles in hybrids $\mathsf{Hyb}_1 \to \mathsf{Hyb}_6$.

Hyb_2 This hybrid replaces the way zero-knowledge proofs are generated: instead of using the proving algorithm $\mathsf{DLEQOR.P}$, we use the zero-knowledge simulator $\mathsf{DLEQOR.Sim}$. We do so for all signing oracles.
If there exists an adversary $\mathsf{A} \in \text{PPT}$ whose output is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of DLEQOR: consider the adversary $\mathsf{B} \in \text{PPT}$ for the game $\mathrm{ZK}^{\beta}_{\mathsf{DLEQOR}}(\lambda)$ that, given as input the group description Γ, generates X_0, X_1 as per $\mathsf{AT.KeyGen}(1^\lambda)$ and then invokes the adversary A. All random oracles queries are performed exactly as per Hyb_2, except for signing queries. In a $\textsc{Sign}$ (and a $\textsc{Sign}_b$) query, after generating the values s, S', and W' as per Hyb_1, the proof π is generated via the $\textsc{Prove}_\beta$ oracle for the statement $((X_0, X_1), T', S', W') \in \mathcal{L}_{2,\Gamma}$. At the end of its execution, A (and so B) return a guess b'.
If the $\textsc{Prove}_\beta$ oracle outputs proofs via $\mathsf{DLEQOR.P}$, the game is identical to Hyb_1, else the game is identical to Hyb_2. It follows that, for any adversary $\mathsf{A} \in \text{PPT}$, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in DLEQOR, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQOR},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda).\right|$$

Hyb_3 We *strengthen* the game: if during any of the signing queries the oracle H_s already had received a query of the form (T', s), we abort. Clearly, the output of the two hybrids is distinguishable only in the case of a collision on the choice of s between the signing oracles, or a collision between the signing oracles themselves. For an adversary $\mathsf{A} \in \text{PPT}$ making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles H_s, $\textsc{Sign}$, or $\textsc{Sign}_b$, the probability that the game aborts is at most $q(q-1)/2p$. It follows that:

$$\frac{q(q-1)}{2p} \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

Hyb_4 We now change the way W' is computed: at key generation phase we sample an additional element $y' \leftarrow_\$ \mathbb{Z}_p$, and for any query to the random oracle $\textsc{Sign}$ we construct W' as $x_b T' + y' S'$ instead of $x_b T' + y_b S'$. The proof π gets simulated as before.
We prove that if, by contradiction, the two games are distinguishable, then there exists an adversary B for the game $\mathrm{DDH}^{\beta}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. The adversary B wins every time the output of the two hybrids is different. The adversary B receives as input a DDH tuple $(P, A := aP, B := bP, C) \in \mathbb{G}^4$ such that $C = abP$ in the case $\mathrm{DDH}^{0}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$ and $C \leftarrow_\$ \mathbb{G}$ in the case $\mathrm{DDH}^{1}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. Given a single challenge (P, A, B, C), B can exploit the random self-reducibility property of DDH to construct q random instances of the DDH challenge: for any $i \leq q$ the adversary B can select $\alpha_i, \beta_i \leftarrow_\$ \mathbb{Z}_p$ and construct the challenge:

$$(P, \quad A, \quad \beta_i B + \alpha_i P, \quad \beta_i C + \alpha_i B)$$

The adversary B proceeds as per Hyb_3, embedding the challenge in the public key and oracle replies. It fixes $H := P$, and instead of generating $X_b := x_b G + y_b H$, it constructs it as $X_b := x_b G + A$. Then, it runs the adversary A. The adversary A will make queries to any of the random oracles H_t and H_s, wheret B programs the RO the response to H_s as we discuss next. We replace queries to the signing oracles:

- for any query $\textsc{Sign}_b(T')$, we sample $s \leftarrow_\$ \{0,1\}^\lambda$ and check for collisions w.r.t. previous queries to H_s as per Hyb_3. Then, we sample $\alpha, \beta \leftarrow_\$ \mathbb{Z}_p$ and we program the random oracle on $\mathsf{H}_s(T', s) = S'$ to reply with $(\beta_i B + \alpha_i P)$, for some $\alpha_i, \beta_i \leftarrow_\$ \mathbb{Z}_p$. Then, B computes $W' := x_b T' + \beta_i C + \alpha_i A$ and produces the proof π using the simulator. B returns (s, W', π)
- for any query $\textsc{Sign}(\hat{b} = b, T')$, after sampling $s \leftarrow_\$ \{0,1\}^\lambda$, we program the random oracle on $\mathsf{H}(T', s) = S'$ to reply with $\alpha_i H$ for some $\alpha_i \leftarrow_\$ \mathbb{Z}_p$. B computes $W' := x_b T' + \alpha_i A$, and simulates the proof. It returns (s, W', π).
- any query to $\textsc{Sign}(\hat{b} = 1 - b, T')$ is handled exactly as per Hyb_3.

At the end of A's execution, B returns whatever guess A returned. We note that if the challenge C is provided according to $\mathrm{DDH}^{0}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, B behaves exactly as per Hyb_3; if the challenge C is provided according to $\mathrm{DDH}^{1}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, B behaves exactly as per Hyb_4. Additionally, if the simulator fails to simulate this statement as it's not in the language, then B also wins the game $\mathrm{DDH}^{\beta}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$. Therefore, every time that A's output is different between the two hybrids (or every time that $\mathsf{DLEQOR.Sim}$ fails), B will distinguish a random tuple from a DDH tuple. It follows therefore that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

Hyb_5 In this game, we remark that $W' := x_b T' + y' S'$, and that $y' \leftarrow_\$ \mathbb{Z}_p$ is used only for computing W'. Therefore, the distribution of W' in Hyb_4 is uniform (plus a constant $x_b T'$, i.e., uniform) as long as $S' \neq 0G$. Therefore, we change once again the way we compute W', swapping b with $1-b$: in this hybrid, $W' := x_{1-b} T' + y' S'$. For the above remarks, the two games can be distinguished only if S' is the identity element, which happens with probability $1/p$.

Hyb_6 In this hybrid, we remove y' and we compute W' using the witness $1 - b$. The proof for this hybrid follows an argument similar to the one used for the transition $\mathsf{Hyb}_3 \rightarrow \mathsf{Hyb}_4$. Using the property of self-reducibility, the adversary would embed the challenge in $X_{1-b} := x_{1-b} T' + B$, and replace the computation of W' in $\textsc{Sign}_b$ with $W' := x_{1-b} T' + \beta_i C + \alpha_i A$ (note: here $S' := \beta \alpha_i + \alpha_i P$). Queries to $\textsc{Sign}(\hat{b} = 1 - b, T')$ are handled in a similar way too: $S' = \alpha_i H$ and $W' := x_{1-b} T' + \alpha_i B$. Therefore, it follows that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

At this point we note that the oracle $\textsc{Sign}_b$ is issuing signatures under the witness x_{1-b}, y_{1-b}. It is possible, through a sequence of hybrids, to remove the condition on the collision of s introduced in Hyb_3 (via the same argument used for the transition $\mathsf{Hyb}_2 \rightarrow \mathsf{Hyb}_3$), and swap back the zero-knowledge simulator with the prover's algorithm $\mathsf{DLEQOR.P}$ (via the same argument used for the transition $\mathsf{Hyb}_1 \rightarrow \mathsf{Hyb}_2$). Therefore, the advantage of an adversary A in winning the game $\mathrm{BL}^{\beta}_{\mathsf{BAT},\mathsf{A}}(\lambda)$

$$\mathsf{Adv}^{\mathrm{hb}}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \frac{q(q-1)}{2^\lambda} + \frac{1}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 2\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQOR}}(\lambda)$$

where q is the number of queries to the signing oracles or to the random oracle H_s and the prime p outputted by GrGen satisfies $\lambda = \lfloor \log_2 p \rfloor$. □

7 Anonymous Tokens with Private Metadata Bit and Validity Verification

In this section, we present a design for an anonymous token that provides both private metadata bit as well as verification validity functionality that can be queried by any party. The basic idea of this design is to combine the token functionality from the previous two sections into one token that has two parts: a token that has no private metadata, which can be used for validity verification, and a second part, which provide a private metadata. It is important that these two parts could not be separated and used independently for the purposed of reading the metadata bit.

Construction 4 *Let $\Gamma := (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be an algorithm that generates a group $\mathbb{G}$ of order p and outputs two distinct random generators G and H. Let $\mathsf{H}_s : \mathbb{G} \times \{0,1\}^* \to \mathbb{G}$ be a random oracle mapping a group element and a string into group elements. Let $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ be a proof system for the DLEQ relationship defining the language*

$$\mathcal{L}_{DLEQ,\Gamma} := \left\{ (X,T,S,W) \in \mathbb{G}^4 \ : \ \exists (x,y) \in \mathbb{Z}_p^2 \ . \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}.$$

Let $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ be a proof system for the DLEQOR relationship defining the language

$$\mathcal{L}_{DLEQOR,\Gamma} := \left\{ (\mathbf{X},T,S,W) \in \mathbb{G}^{n+3} \ : \ \exists (b,x,y) \in \{0,1\} \times \mathbb{Z}_p^2 \ . \ \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}.$$

We construct an anonymous token scheme AT defined by the following algorithms:

- $(\mathsf{sk}, \mathsf{Com}_{\mathsf{sk}}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run $\Gamma := (\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ to obtain group parameters. Γ will be an implicit input to all other algorithms.*
  - *Sample a pair of random values $(\mathbf{x} = (x_0, x_1), \mathbf{y} = (y_0, y_1)) \leftarrow\!\!\$\ \mathbb{Z}_p^2$ and $\tilde{x}, \tilde{y} \leftarrow\!\!\$\ \mathbb{Z}_p$, and set $\mathsf{sk} := (\mathbf{x}, \mathbf{y}, \tilde{x}, \tilde{y})$, $\mathsf{Com}_{\mathsf{sk}} := (\mathbf{X}, \tilde{X})$ where $\mathbf{X} = (x_0 G + y_0 H, x_1 G + y_1 H)$ and $\tilde{X} = \tilde{x} G + \tilde{y} H$.*
- $[\![(t,\sigma), \bot]\!] \leftarrow \langle \mathsf{AT.Usr}(\mathsf{Com}_{\mathsf{sk}}, t), \mathsf{AT.Sig}(\mathsf{Com}_{\mathsf{sk}}, \mathsf{sk}) \rangle$ *– the anonymous token singing protocol is defined in Figure 13.*
- $bool \leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, t, \sigma)$*:*
  - *Parse $\sigma = (S, W, \tilde{W})$.*
  - *If $\tilde{W} = \tilde{x} H(t) + \tilde{y} S$, return 1. Otherwise, return 0.*
- $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, t, \sigma)$*:*
  - *Parse $\sigma = (S, W, \tilde{W})$.*
  - *If $\tilde{W} \neq \tilde{x} H(t) + \tilde{y} S$, return $\bot$.*
  - *Else, if $W = x_0 H(t) + y_0 S$, return 0. If $W = x_1 H(t) + y_1 S$, return 1. Otherwise, return $\bot$.*

The correctness of the above scheme follows from the correctness for its two component tokens.

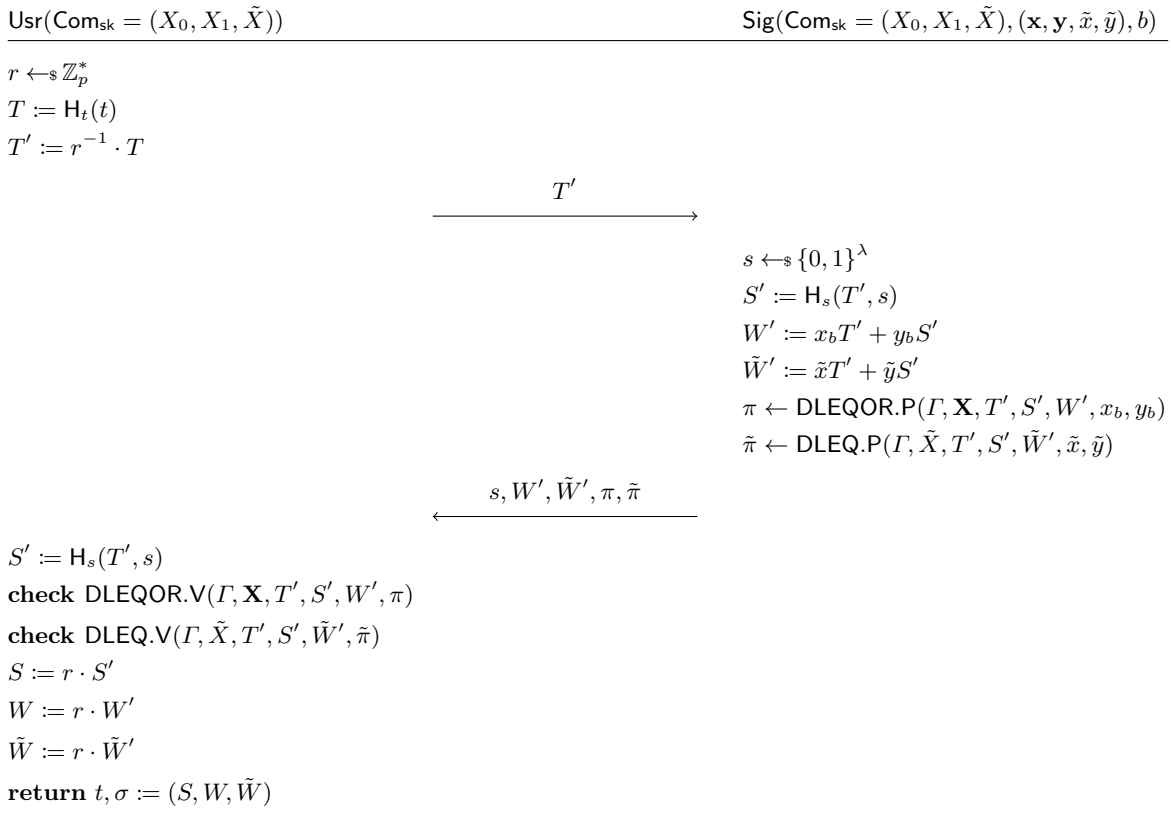

Fig. 13. Issuance protocol for anonymous token with private metadata bit and validity verification.

7.1 Unforgeability

Theorem 16. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a DLEQ zero-knowledge proof system, and* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 4 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

Proof. The one-more unforgeability of this construction follows from the one-more unforgeability of Construction 3. We can construct an adversary for Construction 3 by interacting with an adversary for Construction 4 in the same way as the reduction in the proof of Theorem 13 since the user messages in both constructions are the same, and the server's response in Construction 3 is a subset of the server's response in Construction 4. □

7.2 Unlinkability

Theorem 17. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQ.P}, \mathsf{DLEQ.V})$ *be a DLEQ zero-knowledge proof system, and* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 4 provides* 2*-unlinkability according to Definition 3 assuming the hardness of DDH.*

Proof. The unlinkability of this construction follows from the unlinkability of Construction 3. We can construct an adversary for Construction 3 by interacting with an adversary for Construction 4 in the same way

as the reduction in the proof of Theorem 14 since the user messages in both constructions are the same, and the server's response in Construction 3 is a subset of the server's response in Construction 4. □

7.3 Privacy of Metadata Bit

Theorem 18. *Let* $(\mathbb{G}, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be a parameter generation function such that* H *is a random element in the group (i.e., the discrete log of* H *with respect to* G *is unknown),* $(\mathsf{DLEQOR.P}, \mathsf{DLEQOR.V})$ *be a DLEQOR zero-knowledge proof system. Construction 4 provides privacy for the metadata bit according to Definition 4 assuming the hardness of DDH.*

Proof. The proof for the private metadata bit here follows closely the proof of Theorem 15. The only difference is that we need to handle validity queries from the adversary. However, since the validity is checked only on the part of the token, which is independent of the private metadata bit, the reduction can always have the private parameters for that part of the token and answer the validity oracle query honestly. Figure 14 presents the hybrids for our proof which transition from an execution of of $\mathrm{PMB}^b_{\mathsf{AT},\mathsf{A}}(\lambda)$ to an execution of $\mathrm{PMB}^{1-b}_{\mathsf{AT},\mathsf{A}}(\lambda)$ (see Definition 4). The validity oracle answers are computed in the same way in all hybrids. We argue that each pair of consecutive hybrids are indistinguishable for the adversary and thus $\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{AT},\mathsf{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Hyb_0 This is game $\mathrm{PMB}^b_{\mathsf{AT},\mathsf{A}}(\lambda)$: here, the adversary is provided with the commitment $\mathsf{Com}_{\mathsf{sk}} = (X_0, X_1, \tilde{X})$. The adversary has access to the signing oracle for a bit of its choosing, and a challenge oracle that signs new tokens with the bit b. Additionally, it has access to the various random oracles: H_t, H_s, and H_c. At the end of its execution, it outputs a bit b'.

Hyb_1 In this hybrid, we unroll the procedures for the signing protocol: we generate $s, S', W', \tilde{W}', \pi$ and $\tilde{\pi}$ within the security experiment itself. The two games are perfectly indistinguishable.

Hyb_2 This hybrid replaces the way zero-knowledge proofs for DLEQOR are generated: instead of using the proving algorithm DLEQOR.P, we use the zero-knowledge simulators DLEQOR.Sim. We do so for all signing oracles.
If there exists an adversary $\mathsf{A} \in$ PPT whose output is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of DLEQOR: consider the adversary $\mathsf{B} \in$ PPT for the game $\mathrm{ZK}^b_{\mathsf{DLEQOR}}(\lambda)$ that, given as input the group description Γ, generates $X_0, X_1, \tilde{X}$ as per $\mathsf{Srv.KeyGen}(1^\lambda)$ and then invokes the adversary A. All random oracles queries are performed exactly as per Hyb_2, except for signing queries. In a $\mathsf{AT.Sig}_{\mathsf{sk}}$ (and a $\mathsf{AT.Sig}_{\mathsf{sk},b}$) query, after generating the values s, S', and W' as per Hyb_1, the proof π is generated via the PROVE oracle for the statement $((X_0, X_1), T', S', W') \in \mathcal{L}_{DLEQOR,\Gamma}$. Note that $\tilde{pi}$ is still generated using the prover for the DLEQ proof system. At the end of its execution, A (and so B) return a guess b'.
If the PROVE oracle outputs proofs via DLEQOR.P, the game is identical to Hyb_1, else the game is identical to Hyb_2. It follows that, for any adversary $\mathsf{A} \in$ PPT, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in DLEQOR, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQOR},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda).\right|$$

Hyb_3 We *strengthen* the game: if during any of the signing queries the oracle H_s already had received a query of the form (T', s), we abort. Clearly, the output of the two hybrids is distinguishable only in the case of a collision on the choice of s between the signing oracles, or a collision between the signing oracles themselves. For an adversary $\mathsf{A} \in$ PPT making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles H_s or $\mathsf{AT.Sig}$, the probability that the game aborts is at most $q(q-1)/2p$. It follows that:

$$\frac{q(q-1)}{2p} \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

Hyb_4 We now change the way W' is computed: at key generation phase we sample an additional element $y' \leftarrow\!\!\$\, \mathbb{Z}_p$, and for any query to the random oracle $\mathsf{AT.Sig}$ we construct W' as $x_b T' + y' S'$ instead of $x_b T' + y_b S'$. The proof π gets simulated as before.

We prove that if, by contradiction, the two games are distinguishable, then there exists an adversary B for the game $\mathrm{DDH}^{b}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. The adversary B wins every time the output of the two hybrids is different. The adversary B receives as input a DDH tuple $(P, A \coloneqq aP, B \coloneqq bP, C) \in \mathbb{G}^4$ such that $C = abP$ in the case $\mathrm{DDH}^{0}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$ and $C \leftarrow\!\!\$\, \mathbb{G}$ in the case $\mathrm{DDH}^{1}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. Given a single challenge (P, A, B, C), B can exploit the random self-reducibility property of DDH to construct q random instances of the DDH challenge: for any $i \leq q$ the adversary B can select $\alpha_i, \beta_i \leftarrow\!\!\$\, \mathbb{Z}_p$ and construct the challenge:

$$(P, \quad A, \quad \beta_i B + \alpha_i P, \quad \beta_i C + \alpha_i B).$$

The adversary B proceeds as per Hyb_3, embedding the challenge in the public key and oracle replies. It fixes $H \coloneqq P$, and instead of generating $X_b \coloneqq x_b G + y_b H$, it constructs it as $X_b \coloneqq x_b G + A$. Then, it runs the adversary A. The adversary A will make queries to any of the random oracles H_t, H_s: to these queries we reply exactly as per security experiment; we replace queries to the signing oracles:

- for any query $\mathsf{AT.Sig}_{\mathsf{sk},b}(T')$, we sample $s \leftarrow\!\!\$\, \{0,1\}^\lambda$ and check for collisions w.r.t. previous queries to H_s as per Hyb_3. Then, we program the random oracle on $\mathsf{H}_s(T', s) = S'$ to reply with $(\beta_i B + \alpha_i P)$, for some $\alpha_i, \beta_i \leftarrow\!\!\$\, \mathbb{Z}_p$ that were not used before. Then, B computes $W' \coloneqq x_b T' + \beta_i C + \alpha_i A$ and produces the proof π using the simulator. B computes $\tilde{W}' \coloneqq \tilde{x} T' + \tilde{y}(\beta_i B + \alpha_i P)$ and generates honestly a proof $\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\mathit{\Gamma}, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$. B returns $(s, W', \tilde{W}', \pi, \tilde{\pi})$
- for any query $\mathsf{AT.Sig}_{\mathsf{sk}}(\hat{b} = b, T')$, after sampling $s \leftarrow\!\!\$\, \{0,1\}^\lambda$, we program the random oracle on $\mathsf{H}_s(T', s) = S'$ to reply with $\alpha_i H$ for some $\alpha_i \leftarrow\!\!\$\, \mathbb{Z}_p$. B computes $W' \coloneqq x_b T' + \alpha_i A$, and simulates the proof. B computes $\tilde{W}' \coloneqq \tilde{x} T' + \tilde{y} \alpha_i H$ and generates honestly a proof $\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\mathit{\Gamma}, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$. It returns $(s, W', \tilde{W}', \pi, \tilde{\pi})$.
- any query to $\textsc{Sign}(\hat{b} = 1 - b, T')$ is handled exactly as per Hyb_3.

It is important to note that we can still answer queries to $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot)$ (i.e., verify that a token is valid), as it does not require knowledge of y_b: it only checks the $\tilde{W}$ component of the token. At the end of A's execution, B returns whatever guess A returned. We note that if the challenge C is provided according to $\mathrm{DDH}^{0}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, B behaves exactly as per Hyb_3; if the challenge C is provided according to $\mathrm{DDH}^{1}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, B behaves exactly as per Hyb_4. Additionally, if the simulator fails to simulate this statement as it's not in the language, then B also wins the game $\mathrm{DDH}^{b}_{\mathsf{A},\mathsf{GrGen}}(\lambda)$. Therefore, every time that A's output is different between the two hybrids (or every time that $\mathsf{DLEQOR.Sim}$ fails), B will distinguish a random tuple from a DDH tuple. It follows therefore that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

Hyb_5 In this game, we remark that $W' \coloneqq x_b T' + y' S'$, and that $y' \leftarrow\!\!\$\, \mathbb{Z}_p$ is used only for computing W'. Therefore, the distribution of W' in Hyb_4 is uniform (plus a constant $x_b T'$, i.e. uniform) as long as $S' \neq 0G$. Therefore, we change once again the way we compute W', swapping b with $1-b$: in this hybrid, $W' \coloneqq x_{1-b} T' + y' S'$. For the above remarks, the two games can be distinguished only if S' is the identity element, which happens with probability $1/p$.

Hyb_6 In this hybrid, we remove y' and we compute W' using the witness $1 - b$. The proof for this hybrid follows an argument similar to the one used for the transition $\mathsf{Hyb}_3 \to \mathsf{Hyb}_4$. Using the property of self-reducibility, the adversary would embed the challenge in $X_{1-b} \coloneqq x_{1-b} T' + B$, and replace the computation of W' in $\mathsf{AT.Sig}_{\mathsf{sk},b}$ with $W' \coloneqq x_{1-b} T' + \beta_i C + \alpha_i A$ (note: here $S' \coloneqq \beta \alpha_i + \alpha_i P$). The values $\tilde{W}'$ and $\tilde{\pi}'$ are computed honestly. Queries to $\mathsf{AT.Sig}_{\mathsf{sk}}(\hat{b} = 1 - b, T')$ are handled in a similar way too: $S' = \alpha_i H$ and $W' \coloneqq x_{1-b} T' + \alpha_i B$. Therefore, it follows that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

At this point we note that the oracle $\mathsf{AT.Sig}_{\mathsf{sk},b}$ is issuing signatures under the witness x_{1-b}, y_{1-b}. It is possible, through a sequence of hybrids, to remove the condition on the collision of s introduced in Hyb_3 (via the same argument used for the transition $\mathsf{Hyb}_2 \to \mathsf{Hyb}_3$), and swap back the zero-knowledge simulator with

the prover's algorithm DLEQOR.P (via the same argument used for the transition $\mathsf{Hyb}_1 \to \mathsf{Hyb}_2$). Therefore, the advantage of an adversary A in winning the game $\mathrm{PMB}^b_{\mathsf{AT},\mathsf{A}}(\lambda)$

$$\mathsf{Adv}^{\mathrm{hb}}_{\mathsf{BAT},\mathsf{A}}(\lambda) \leq \frac{q(q-1)}{2^\lambda} + \frac{1}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 2\mathsf{Adv}^{\mathrm{zk}}_{\mathsf{DLEQOR}}(\lambda)$$

where q is the number of queries to the signing oracles or to the random oracle H_s and the prime p outputted by GrGen satisfies $\lambda = \lfloor \log_2 p \rfloor$. □

Oracle $\mathsf{AT.Sig}_{\mathsf{sk}}(\hat{b} = b, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$

$s \leftarrow\!\!\$ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\tilde{W}' := \tilde{x} T' + \tilde{y} S'$
$\pi \leftarrow \mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, T', S', W', x_{\hat{b}}, y_{\hat{b}})$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
$\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\Gamma, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$
return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

Oracle $\mathsf{AT.Sig}_{\mathsf{sk}}(\hat{b} = 1 - b, T')$ in $\mathsf{Hyb}_4(\lambda), \mathsf{Hyb}_5(\lambda), \mathsf{Hyb}_6(\lambda)$

$s \leftarrow\!\!\$ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\tilde{W}' := \tilde{x} T' + \tilde{y} S'$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
$\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\Gamma, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$
return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

Oracle $\mathsf{VerValid}(t, \sigma := (S, W, \tilde{W}))$ in all $\mathsf{Hyb}_1(\lambda) - \mathsf{Hyb}_6(\lambda)$

If $\tilde{W} = \tilde{x} H(t) + \tilde{y} S$, $bool = 1$. Otherwise, $bool = 0$.
return $bool$

Oracle $\mathsf{AT.Sig}_{\mathsf{sk},b}(T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$

$s \leftarrow\!\!\$ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$\tilde{W}' := \tilde{x} T' + \tilde{y} S'$
$\pi \leftarrow \mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, T', S', W', x_{\hat{b}}, y_{\hat{b}})$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
$\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\Gamma, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$
return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

Oracle $\mathsf{AT.Sig}_{\mathsf{sk},b}(T')$ in $\mathsf{Hyb}_4(\lambda)$, $\mathsf{Hyb}_5(\lambda)$, $\boxed{\mathsf{Hyb}_6(\lambda)}$

$s \leftarrow\!\!\$ \{0,1\}^\lambda$
if $\exists$query $\mathsf{H}_s(T', s)$: **abort**
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$W' := x_b T' + y' S'$
$W' := x_{1-b} T' + y' S'$
$\boxed{W' := x_{1-b} T' + y_{1-b} S'}$
$\tilde{W}' := \tilde{x} T' + \tilde{y} S'$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W', x_b, y_b)$
$\pi \leftarrow \mathsf{DLEQOR.Sim}(\Gamma, \mathbf{X}, T', S', W')$
$\tilde{\pi} \leftarrow \mathsf{DLEQ.P}(\Gamma, \tilde{X}, T', S', \tilde{W}', \tilde{x}, \tilde{y})$
return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

Fig. 14. Summary of the proof for private metadata bit, with highlighted changes to the signing oracles in hybrids $\mathsf{Hyb}_1 \to \mathsf{Hyb}_6$.

Acknowledgments

The authors thank Fabrice Benhamouda, Charlie Harrison, Michael Kleber, Steven Valdez, and Moti Yung for helpful discussions.

References

ABR01. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

BG93. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 390–420. Springer, Heidelberg, August 1993.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

BP02. Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Heidelberg, August 2002.

CA89. David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1989.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

Cha90. David Chaum. Zero-knowledge undeniable signatures. In *EUROCRYPT*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer, 1990.

CKS09. David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009.

DGS+18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

ECS+15. Adam Everspaugh, Rahul Chatterjee, Samuel Scott, Ari Juels, and Thomas Ristenpart. The pythia PRF service. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 547–562, 2015.

Hen14. Ryan Henry. Efficient zero-knowledge proofs and applications. 2014.

HL06. Javier Herranz and Fabien Laguillaumie. Blind ring signatures secure under the chosen-target-CDH assumption. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 117–130. Springer, Heidelberg, August / September 2006.

JKK14. Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253. Springer, Heidelberg, December 2014.

JKX18. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486. Springer, Heidelberg, April / May 2018.

JL10. Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 418–435. Springer, Heidelberg, September 2010.

KM08. Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and diffie-hellman problems. *J. Mathematical Cryptology*, 2(4):311–326, 2008.

LQ04. Benoît Libert and Jean-Jacques Quisquater. The exact security of an identity based signature and its applications. Cryptology ePrint Archive, Report 2004/102, 2004. http://eprint.iacr.org/2004/102.

Oka93. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

Game $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$

$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$(x, X) \leftarrow \mathsf{Elg.KeyGen}(\Gamma)$
for $i \in [\ell+1]$:
  $M_i \leftarrow\!\!\$\ \mathbb{G}$
  $c_i \leftarrow \mathsf{Elg.Enc}(X, M_i)$
$(M_i^*)_i^\ell = \mathsf{A}^{\mathsf{Elg.Dec}^{(\ell)}(x,\cdot)}(X, c_0, \ldots, c_\ell)$
return $M_i^* = M_i$ for all $i \in [\ell+1]$

Adversary $\mathcal{B}(\Gamma, X)$ for $\text{CTDH}_{\mathsf{GrGen},\mathcal{B}}(\lambda)$

for $i \in [\ell+1]$:
  $Y_i \leftarrow \textsc{Target}(i)$; $R_i \leftarrow\!\!\$\ \mathbb{G}$
  $c_i := (C_i, D_i) := (Y_i, Y_i + R_i)$
// $\textsc{Dec}(i) := Y_i + R_i - \textsc{Help}(Y_i)$
$(M_0, \ldots, M_\ell) \leftarrow \mathsf{A}^{\textsc{Dec}}(X, c_0, \ldots, c_\ell)$
for $i \in [\ell+1]$: $Z_i := Y_i + R_i - M_i$
return $(i, Z_i)_{i=0}^\ell$

Fig. 15. Game for one-more decryption (left) and reduction to chosen-target Diffie-Hellman (right).

OP01. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, Heidelberg, February 2001.

A Equivalence of CTDH and 1MD

Privacy Pass [DGS+18] is proven unforgeable under one-more decryption security of Elgamal. One-more decryption security states that it is difficult for any PPT adversary A to decrypt $\ell+1$ Elgamal ciphertexts of random messages, even when given access to an oracle for ℓ Elgamal encryptions.

Elgamal (denoted Elg) is a public-key cryptosystem based on a group generator algorithm GrGen. Elgamal achieves semantic security if DDH holds, i.e., if $\mathsf{Adv}^{\text{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is negligible. The key generation algorithm $\mathsf{Elg.KeyGen}(\Gamma)$ outputs a pair $(\mathsf{sk}, \mathsf{pk}) := (x, X := xG)$ where $x \leftarrow\!\!\$\ \mathbb{Z}_p$. The encryption algorithm $\mathsf{Elg.Enc}(M)$ outputs a ciphertext $(C := cG, D := cX + M)$ where $c \leftarrow\!\!\$\ \mathbb{Z}_p$. The decryption algorithm $\mathsf{Elg.Dec}(x, (C, D))$ takes as input the secret key and the ciphertext, and otuputs the message $M = D - xC$.

Theorem 19. *Chosen-Target Diffie-Hellman for* GrGen *holds if and only if One-More Decryption Security for* $\mathsf{Elg}[\mathsf{GrGen}]$ *holds. In other words,* $\text{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ *is equivalent to* $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}]}(\lambda)$

Proof. We start by proving that 1MD $\implies$ CTDH. Let $\mathsf{A} \in$ PPT be an adversary for $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$. We use it to construct an adversary $\mathcal{B}$ for the game $\text{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$. The adversary $\mathcal{B}$ receives as input a group drescription Γ and a group element $X \in \mathbb{G}$. Additionally, it has access to two oracles: a $\textsc{Target}$ oracle and a $\textsc{Help}$ oracle. It start by querying the $\textsc{Target}$ oracle $\ell+1$ times on $i \in [\ell+1]$, thus receiving $\textsc{Target}(i) = Y_i \in \mathbb{G}$. It samples uniformly at random $R_i \leftarrow\!\!\$\ \mathbb{G}$ for all $i \in [\ell+1]$ and invokes $\mathsf{A}(X, (C_i, D_i)_i^\ell)$, where $(C_i, D_i) := (Y_i, Y_i + R_i)$. During its execution, the adversary A may ask for ℓ decryption queries. We answer to those queries returning $M_i := (Y_i + R_i) - Z_i$ where $Z_i := \textsc{Help}(Y_i)$. At the end of its execution, the adversary $\mathcal{B}$ returns $\ell+1$ decryption $(M_0, \ldots, M_\ell) \in \mathbb{G}^{\ell+1}$. The adversary $\mathcal{B}$ returns $(i, Z_i := Y_i + R_i - M_i)$ for each $i \in [\ell+1]$.

The distribution of each ciphertext is uniform over $\mathbb{G}^2$, exactly as in the game $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$ (because each message is uniformly random in $\mathbb{G}$). Decryption queries are responded exactly in the same way (subtracting off the CDH of the randomness Y_i with the public key X). Furthermore, if the adversary wins the game $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$, it must be the case that it returned M_i for all $i \in [\ell+1]$ such that:

$$M_i = \mathsf{Elg.Dec}(x, (C_i, D_i)) \implies Z_i := (Y_i + R_i) - M_i = \mathsf{CDH}(X, Y_i)$$

The adversary $\mathcal{B}$ wins the game $\text{CTDH}_{\mathsf{GrGen},\mathcal{B}}(\lambda)$ every time that the adversary A wins the game $1\text{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$, therefore for all $\mathsf{A} \in$ PPT:

$$\mathsf{Adv}^{\text{ctdh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) \geq \mathsf{Adv}^{\text{1md}}_{\mathsf{GrGen},\mathcal{B}}(\lambda).$$

We now prove the reverse implication, that is, CTDH $\implies$ 1MD. Let $\mathsf{A} \in$ PPT be an adversary for $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$. We use it to construct an adversary $\mathcal{B}$ for $\mathrm{1MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A}}(\lambda)$.

The adversary $\mathcal{B}$ receives as input a group description Γ together with a group element $X = xG \in \mathbb{G}$ (for some secret $x \in \mathbb{Z}_p$) and a sequence of $\ell+1$ ciphertexts $c_0, \ldots, c_\ell$ such that $c_i = (C_i, D_i)$ for all $i \in [\ell+1]$. $\mathcal{B}$ starts off by selecting a random into map $\mu : \{0,1\}^\lambda \to \mathbb{Z}_p$. Then, it invokes the adversary A on (Γ, X), overriding random oracle queries in the following way:

- for any query of the form $\textsc{Target}(t)$, the adversary $\mathcal{B}$ returns $Y = \sum_{j=0}^{\ell} a^{\mu(t)j} C_i$;
- for any query of the form $\textsc{Help}(C)$, the adversary $\mathcal{B}$ samples $D \leftarrow_{\$} \mathbb{G}$, and queries the decryption oracle $\mathsf{Dec}((C,D))$, thus obtaining $M = \mathsf{Elg.Dec}(x,(C,D)) = R - \mathsf{CDH}(X,Y)$. It returns $R - M = \mathsf{CDH}(X,Y)$.

At the and of its execution, the adversary A returned $\ell+1$ pairs (t_i, Z_i). By winning condition of $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$, for each $i \in [\ell+1]$, Z_i satisfies:

$$x \cdot \begin{bmatrix} Y_0 \\ \vdots \\ Y_\ell \end{bmatrix} = x \cdot \begin{bmatrix} a^{0\cdot\mu(t_0)} & \ldots & a^{\ell\cdot\mu(t_0)} \\ a^{0\cdot\mu(t_1)} & \ldots & a^{\ell\cdot\mu(t_1)} \\ \vdots & \ddots & \vdots \\ a^{0\cdot\mu(t_\ell)} & \ldots & a^{\ell\cdot\mu(t_\ell)} \end{bmatrix} \begin{bmatrix} C_0 \\ \vdots \\ C_\ell \end{bmatrix} = \begin{bmatrix} Z_0 & \cdots & Z_{\ell+1} \end{bmatrix}.$$

The matrix $A \coloneqq [a^{j\mu(t_i)}]_{i,j}$ is a Vandermonde matrix. The second winning condition states that $t_i \neq t_j$ for all $i, j \in [\ell+1]$, $i \neq j$. Therefore, $a^{\mu(t_i)} \neq a^{\mu(t_j)}$. Therefore, A is invertible for all $i \neq j$. Let A' be the inverse of A; then, $Z'_i = \sum_j a'_{i,j} Z_i = \mathsf{CDH}(X, C_i)$. The adversary $\mathcal{B}$ returns $M_i \coloneqq D_i - Z_i$, for all $i \in [\ell+1]$.

The replies to the $\textsc{Help}$ oracle are identical to the ones in $\mathrm{CTDH}_{\mathsf{GrGen},\mathcal{B}}(\lambda)$. The replies to the $\textsc{Target}$ oracle follow the uniform distribution in $\mathbb{G}$ too, since μ is a random injective map hidden from the view of the adversary.

It follows that the adversary $\mathcal{B}$ wins the game $\mathrm{1MDH}_{\mathsf{Elg}[\mathsf{GrGen}],\mathcal{B}}(\lambda)$ every time that the adversary A wins the game $\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$, therefore for all $\mathsf{A} \in$ PPT:

$$\mathsf{Adv}^{\mathrm{ctdh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{1md}}_{\mathsf{GrGen},\mathcal{B}}(\lambda).$$

□

B Okamoto-Schnorr DLEQOR Proofs

In the section we provide the constructions for the DLEQ and DLEQOR proofs that we use for our token constructions. Given a group description $\Gamma \coloneqq (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ and $n = \mathsf{poly}(\lambda)$, we provide a zero-knowledge argument of knowledge for the following language:

$$\mathcal{L}_{2,\Gamma} \coloneqq \left\{ (\mathbf{X}, T, S, W) \in \mathbb{G}^{n+3} \;:\; \exists (b, x, y) \in [n] \times \mathbb{Z}_p^2 \;.\; \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}$$

Note that when $y = 0$ and $n = 1$ the language is the same of Privacy Pass [DGS+18]. Note that for $n = 1$ the above language is Okamoto-Schnorr [Oka93] for the DLEQ relation.

We present our construction in Figure 16. The setup algorithm generates the group description $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$, and instantiates a random oracle H_c that maps any sequence of elements $(\Gamma, G_1, \ldots, G_n) \in \Gamma \times \mathbb{G}^n$ to a scalar $c \in \mathbb{Z}_p$. The prover and verifier algorithms are the OR-composition of Okamoto-Schnorr [Oka93]. We present the non-interactive version of the protocol, after applying the Fiat-Shamir reduction, for practical convenience.

The proving algorithm (Fig. 16, left) proceeds simulating the transcripts for all $i \in [n], i \neq b$, and choosing the challenges uniformly at random, constrained that their sum is the challenge provided by H_c.

$\mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, T, S, W), (x_b, y_b))$

$(\mathbb{G}, p, G, H) \coloneqq \Gamma$
$k_0, k_1 \leftarrow_\$ \mathbb{Z}_p^2$
$\mathbf{K}_b \coloneqq k_0 \cdot (G;T) + k_1 \cdot (H;S)$
for $i \in [n], \quad i \neq b$
  $c_i, u_i, v_i \leftarrow_\$ \mathbb{Z}_p$
  $\mathbf{K}_i \coloneqq u_i \cdot (G;T) + v_i \cdot (H;S) - c_i \cdot (X_i;W)$
$c \coloneqq \mathsf{H}_c(\Gamma, \mathbf{X}, T, S, W, \mathbf{K}_0, \ldots, \mathbf{K}_n)$
$c_b \coloneqq c - \sum_{i \neq b} c_i$
$u_b \coloneqq k_0 + c_b x_b$
$v_b \coloneqq k_1 + c_b y_b$
return $(\mathbf{c}, \mathbf{u}, \mathbf{v})$

$\mathsf{DLEQOR.V}(\Gamma, \mathbf{X}, S, T, W, (\mathbf{c}, \mathbf{u}, \mathbf{v}))$

$\forall i \in [n]: \quad \mathbf{K}_i \coloneqq u_i(G;T) + v_i \cdot (H;S) - c_i(X_i;W)$
$c \coloneqq \sum_i c_i$
return $c = \mathsf{H}_c(\Gamma, \mathbf{X}, T, S, W, \mathbf{K}_0, \ldots, \mathbf{K}_n)$

Fig. 16. Okamoto-Schnorr proof for DLEQOR.

$\mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W}, x, y)$

$e_0, \ldots, e_m \coloneqq \mathsf{H}(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W})$
$\overline{T} \coloneqq \sum_j^m e_j T_j$
$\overline{S} \coloneqq \sum_j^m e_j S_j$
$\overline{W} \coloneqq \sum_m^m e_j W_j$
return $\mathsf{DLEQOR.P}(\Gamma, \mathbf{X}, \overline{T}, \overline{S}, \overline{W}, x, y)$

$\mathsf{DLEQOR.V}(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W}, \pi)$

$e_0, \ldots, e_m \coloneqq \mathsf{H}(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W}, \pi)$
$\overline{T} \coloneqq \sum_j^m e_j T_j$
$\overline{S} \coloneqq \sum_j^m e_j S_j$
$\overline{W} \coloneqq \sum_m^m e_j W_j$
return $\mathsf{DLEQOR.V}(\Gamma, \mathbf{X}, \overline{T}, \overline{S}, \overline{W}, \pi)$

Fig. 17. Batched Okamoto-Schnorr proof for DLEQOR.

The verification algorithm (Fig. 16, right) checks the validity of all transcripts, and that the sum of the challeges $\sum_i c_i = c$ is the hash of the commitments $\mathsf{H}_c(\Gamma, \mathbf{X}, T, S, W, \mathbf{K}_0, \ldots, \mathbf{K}_n)$. When $n = 1$, we will use the notation $\mathsf{DLEQ}(\Gamma, X, T, S, W)$ to explicit the protocol is using only one public key.

The protocol has special soundness by standard or-composition of sigma protocols [CDS94]: from two transcripts $(\mathbf{c}, \mathbf{u}, \mathbf{v})$ and $(\mathbf{c}', \mathbf{u}', \mathbf{v}')$ verifying simultaneously, some $b \in [n]$ such that $c_b \neq c'_b$ it is possible to extract a witness (x_b, y_b) by computing $x_b \coloneqq (u_b - u'_b)/(c_b - c'_b)$ and $y_b \coloneqq (v_b - v'_b)/(c_b - c'_b)$.

The *(non-interactive Fiat-Shamir reduction of the)* protocol is also zero-knowledge: the simulator simply produces valid transcripts for all $i \in [n]$ by selecting $c_i, u_i, v_i \leftarrow_\$ \mathbb{Z}_p$, and computing $\mathbf{K}_i \coloneqq u_i(G;T) + v_i(H;S) - c_i(X_i, W)$. Then, it computes $c \coloneqq \sum_i^n c_i$ and programs the random oracle to reply with c when queried on $(\Gamma, \mathbf{X}, T, S, W, \mathbf{K}_0, \ldots, \mathbf{K}_n)$. The simulator aborts if such a query was already made, which since K_i are all distributed randomly happens with probability at most $q(\lambda)/p^n$, where $q(\lambda)$ is an upper-bound on the number of queries of the adversary to H_c.

Batching. The proof can be batched via the same technique of Henry [Hen14]: it is possible to prove knowledge of a witness (b, x, y) for m different statements $(\mathbf{X}, T_j, S_j, W_j)_{j=0}^m$ with a single proof.

Theorem 20. *Assuming the discrete logarithm is hard for* GrGen*, the* DLEQOR *proof system depicted in Fig. 17 is a zero-knowledge argument of knowledge.*

Proof (Knowledge soundness). Knowledge soundness for the batched protocol follows from the knowledge soundness of the underlying DLEQ proof system, except for a small statistical error. By soundness of the

underlying proof system we have that, except with a negligible extraction error,

$$\mathsf{DLEQ}(\Gamma, \mathbf{X}, \overline{T}, \overline{S}, \overline{W}, \pi) = \mathbf{true} \;\text{ implies }\; \exists (b, x, y) \;.\; \big((\mathbf{X}, \overline{T}, \overline{S}, \overline{W}), (b, x, y)\big) \in \mathsf{R}(\mathcal{L}_{\Gamma,n}),$$

where $\mathsf{R}(\mathcal{L}_{n,\Gamma})$ denotes the relation associated to the language $\mathcal{L}_{n,\Gamma}$. In other words, there exists an PPT extractor that outputs b, x, y such that $X_b = xG + yH$ and $\overline{W} = x\overline{T} + y\overline{S}$. The values $\overline{T}, \overline{S}$, and $\overline{W}$ are a random linear combination of the elements $(T_i, S_i, W_i)_i^m$. We prove by induction on m that the probability that there exists any $i \in [m]$ such that the (batched) protocol verifies and $W_i \neq xT_i + yS_i$ is at most $2(m+1)/p = \mathsf{negl}(\lambda)$. Logically, it will follow that, except with negligible probability, $(b, x, y) \in [n] \times \mathbb{Z}_p \times \mathbb{Z}_p$ is a valid witness for the statement $(\mathbf{X}, T_i, S_i, W_i) \in \mathcal{L}_{n,\Gamma}$, for each $i \in [m]$.

If $m = 1$, then $\overline{W} = x\overline{T} + y\overline{S}$ can be written as $e_0 W_0 = e_0(xT_0 + yS_0)$. Therefore, $W_0 = xT_0 + yS_0$ iff $e_0 \neq 0$, which happens with probability $1/p < 3/p$. Let us denote with $\Pr[E_m]$ the probability that the (batched) verification equation is satisfied, but the witness is invalid for at least one of the m statements. Note that for the case $\Pr[E_{m+1}]$ there are two possibilities: either $W_m = xT_m + yS_m$, in which case we are left with the equation of the inductive step:

$$\sum_j^{m-1} e_j W_j = \left(x \sum_j^{m-1} e_j T_j + y \sum_j^{m-1} e_j S_j \right),$$

Alternatively, if $W_m \neq xT_m + yS_m$, then either the coefficient e_m is zero or also the other statement must be invalid. It follows that:

$$\Pr[E_{m+1}] \leq \Pr[E_m] + \frac{1}{p}(1 - \Pr[E_m]) + \frac{p-1}{p} \Pr[E_m] \,.$$

(In fact, if $e_m = 0$ we fall in the inductive case; and the probability that the verification equation is invalid is at least $1 - \Pr[E_m]$.) It follows that $\Pr[E_{m+1}] \leq 2\Pr[E_m] - 2/p \Pr[E_m] + 1/p \leq 2\Pr[E_m] + 1/p \leq 2(m+1)/p$. Thus:

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQOR}_{\mathrm{batched}}}(m, \lambda) \leq \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQOR}_{\mathrm{simple}}}(\lambda) + \frac{2(m+1)}{p}$$