Efficient Anonymous Tokens with Private Metadata Bit

Ben Kreuter[1], Tancrède Lepoint[1], Michele Orrù[234], and Mariana Raykova[1]

[1] Google, {benkreuter,tancrede,marianar}@google.com
[2] École Normale Supérieure, CNRS, PSL University, Paris, France, michele.orru@ens.fr
[3] Inria, Paris, France
[4] Recurse Center, New York, USA

March 24, 2020

Abstract. We present a cryptographic construction for anonymous tokens with private metadata bit, called PMBTokens. This primitive enables an issuer to provide a user with anonymous trust tokens that can embed a single private bit, which is accessible only to the party who holds the secret authority key and is private with respect to anyone else. Our construction extends the functionality of Privacy Pass (PETS 2018) with this private metadata bit capability. It provides unforgeability, unlinkability, and privacy for the metadata bit properties based on the DDH and CTDH assumptions in the random oracle model.
Both Privacy Pass and PMBTokens rely on discrete logarithm proofs (DLog equality, or OR of DLog equality). We present techniques to remove the need for proofs of knowledge in these constructions and achieve a slightly weaker notion of unlinkability.
We implement our constructions and we report their efficiency costs.

1 Introduction

The need to propagate trust signals while protecting anonymity has motivated cryptographic constructions for anonymous credentials [14, 12]. While we have constructions that support proofs of complex statements about the owner of the anonymous credential, this comes with computation and communication costs. On the other hand, some practical uses require very simple functionality from the anonymous credential while having very strict efficiency requirements. One such example is the setting for Privacy Pass [22]. Privacy Pass was designed as a tool for content delivery networks (CDNs), which need a way to distinguish honest from malicious content requests, so as to block illegitimate requests that create malicious traffic, drain network resources, and could be used for denial of service attacks. Previous solutions leveraged IP reputation to assess the reputation of users. While helpful in many cases, IP reputation may also lead to a high rate of false positives because of shared IP use. In particular, this introduces unfair burden for users of Tor and VPNs in the context of CDNs. Privacy Pass [22] proposes a solution for this problem: using anonymous tokens as a mechanism to prove trustworthiness of the requests without compromising the privacy of the user. Since CDNs need to potentially handle millions of requests per second, efficiency of the cryptographic construction is of an extreme importance.

In this paper, we consider the functionality of anonymous tokens that enable to convey one of two types of trust signals in a way that the user cannot distinguish which of the two signals is embedded in her tokens. This extension is motivated by the fact that in a system relying on anonymous trust tokens, malicious users who have been identified as such, will become aware of this fact if the issuer stops providing them with tokens. However, since real world attackers have means to corrupt honest users, finding out when they have been detected could serve as an incentive for them to corrupt more users. Being able to pass on the information whether a user is whitelisted or blacklisted, and consume it in appropriate ways on the authentication side, mitigates such behavior.

More concretely, we consider an anonymous token primitive that provides the following functionality: a user and an issuer interact and as a result of this interaction, the user obtains a trust token with a private metadata bit (PMB) embedded in it. The private metadata bit can be read from a token using the secret key for the scheme when the user redeems that token with the issuer. Each token is one-time use, which enables the issuer to update the trust assigned to each user without requiring a complex revocation process by just adjusting the number of tokens that can be issued at once and the frequency of serving new token requests. This anonymous token scheme offers the following security properties: unforgeability, unlinkability, and privacy of the metadata bit. *Unforgeability* guarantees that nobody but the issuer, who holds the secret key, can generate new valid tokens. *Unlinkability* in the presence of the private metadata bit guarantees that the tokens that were issued with the same private metadata bit are indistinguishable to the issuer when redeemed. *Privacy of the metadata bit* states that no party that does not have the secret key can distinguish any two tokens, including tokens issued with different metadata bits.

Our goal is to construct a primitive which achieves the above properties, and has competitive efficiency introducing minimal overhead over Privacy Pass.

1.1 Our Contributions

We present PMBTokens, a construction for anonymous tokens with private metadata bit with communication less than twice the communication of Privacy Pass and computation three times the computation of Privacy Pass, which improves significantly over general techniques that could be used to solve the problem. The security of our constructions relies on the decisional Diffie–Hellman (DDH) and chosen-target Diffie–Hellman (CTDH) problems, and is proven in the random oracle model (ROM). We further show how to simplify and optimize both the construction of Privacy Pass and PMBTokens by using a weaker notion of unlinkability and removing the discrete log equality proofs (DLEQ) and OR of discrete log equality proofs (DLEQOR) from the constructions.

Privacy Pass and Its Challenges to Private Metadata. Our starting point is the Privacy Pass construction which uses the VOPRF primitive [32] $F_k(t) = k\mathsf{H}_t(t)$ that offers an oblivious evaluation mechanism where the user sends to the issuer $r\mathsf{H}_t(t)$ for a randomly selected value r, receives back $rk\mathsf{H}_t(t)$ from which she recovers $k\mathsf{H}_t(t)$.[5]

The above construction does not suffice for unlinkability since the issuer could use a different VOPRF key for each user, which would enable him to fingerprint users. To avoid this issue, Privacy Pass requires the issuer to publish a public parameter constructed with his secret key and later provides a DLEQ proof that proves that she has used the same key to construct the public parameter and the evaluated token. Once all tokens are guaranteed to be issued with the public key, unlinkability follows from the blinding factor r which makes the distribution of $r\mathsf{H}_t(t)$ uniform even when knowing t. The DLEQ proof guarantees that the token value is $x\mathsf{H}_t(t)$, which is independent of the blinding factor.

There is a natural idea to upgrade the above functionality to support a private metadata bit, which is to use two secret keys and use each of these keys for one of the bits. However, this idea does not work directly; the reason for this stems from the fact that the underlying VOPRF is a deterministic primitive. In particular, this means that if we are using two different VOPRF keys for tokens issued with different private metadata bit values, the VOPRF evaluations on the same input t will be the same if they are issued with the same key and will be different with high probability if used with different keys. Thus, if the user obtains multiple

[5] In Privacy Pass the resulting value $k\mathsf{H}_t(t)$ is used for the derivation of a HMAC key in order to avoid credential hijacking. To simplify our presentation, we skip this step and simply assume that credentials are redeemed over a secure channel.

tokens using the same input value t, she will be able to distinguish which ones where issued with the same bit.

New Randomized Tokens and Private Metadata. To resolve the above issue we introduce a construction which makes the token issuance a randomized functionality where the randomness is shared between the user and the issuer. We use the following function $F_{K:=(x,y)}(t;S) = x\mathsf{H}_t(t)+yS$, where t is the value that will be input of the user and S is the randomness of the evaluation, which will be determined by the two parties, more specifically $S = r^{-1}\mathsf{H}_s(r\mathsf{H}_t(t);s)$ where r is the blinding factor chosen by the user and s is a random value chosen by the issuer. This functionality suffices to construct a new anonymized token functionality where during the oblivious evaluation the user sends $T' = r\mathsf{H}_t(t)$, receives back from the issuer $s, W' = xT' + y\mathsf{H}_s(T';s)$, unblinds the values $S = r^{-1}\mathsf{H}_s(T';s)$ and $W = r^{-1}W'$, and outputs a token (t, S, W). The token verification checks that $W = x\mathsf{H}_t(t) + yS$. In order to provide verifiability, the issuer provides an element of the form $X = xG + yH$ and sends a Okamoto–Schnorr DLEQ proof [39, 43] for the statement that $X = xG + yH$ and $W = x\mathsf{H}_t(t) + yS$ are computed using the same secret key (x, y). This functionality is similar to the blind Okamoto–Schnorr signatures [39], with the key difference that we redefine this as a secret key primitive which enables us to have a round-optimal blind evaluation algorithm.

We apply the idea of using two different keys for each private metadata bit value to the above randomized construction; the resulting construction is called PMBTokens. The public parameters are now a pair $(X_0 \coloneqq x_0G+y_0H, X_1 \coloneqq x_1G + y_1H)$, a token issued with a private metadata bit b is of the form $W' = x_b\mathsf{H}_t(t) + y_bS$ and the DLEQ proof is replaced with a DLEQOR proof stating that either W' and X_0, or W' and X_1, are computed using the same secret key (x_0, y_0) or (x_1, y_1).

Removing Discrete Log Proofs. Both Privacy Pass and PMBTokens employ zero-knowledge arguments of knowledge to achieve unlinkability. This approach guarantees that the user can verify that she has obtained a token issued under the same secret key as in the issuer's public parameters. Unlinkability follows from the fact that tokens issued under the same secret key are indistinguishable. We consider a slightly weaker guarantee for the user from the token issuance, which is that either the token she has received is issued under the public key or the token is indistinguishable from a random value, however, the user cannot distinguish these two cases. The implication of this weaker issuance guarantee for unlinkability is that the issuer can distinguish correctly issued tokens from incorrectly issued tokens. Note that in any of the above constructions the issuer also distinguishes valid from invalid, however, the difference is that the verifiability property on the user side also enables her to distinguish whether her tokens are valid. With the weaker issuance guarantee of above, a user will not be able to know in advance whether she has a token that will be valid at redemption. Another difference is that incorrectly formed tokens issued during an execution where the issuer misbehaves, will be indistinguishable for the issuer from incorrectly formed tokens that a malicious user may try to use.

We present modifications of both Privacy Pass and PMBTokens that satisfy this version of unlinkability, while removing the need for DLEQ or DLEQOR proofs and improving the computational cost for the issuer, which is the bottleneck in systems that need to support large number of users who perform many transactions and hence need to obtain tokens regularly.

Our approach for removing the DLEQ proof from Privacy Pass borrows ideas from the construction of a verifiable partially oblivious PRF of Jarecki et al. [33], but simplifies their construction which has additional complexity in order to achieve user verifiability. We use the idea to use not only multiplicative but also additive blinding of the user's input in the form $T' = r(\mathsf{H}_t(t) - \rho G)$. Now, an honest evaluation of the issuer $W' = xr(\mathsf{H}_t(t) - \rho G)$ can be unblinded by the user by computing $r^{-1}W + \rho X = x\mathsf{H}_t(t) - \rho(xG) + \rho X = x\mathsf{H}_t(t)$, where $X = xG$ is the issuer's public key. On the other hand, any dishonestly computed W' which is of the form $W' = r^{-1}T' + P$ for some $P \neq 0$ when unblinded will contain a random additive factor $r^{-1}P$, thus the resulting value will be random. Similarly to Jarecki et al. [33], we can recover verifiability by doing another oblivious evaluation on the same value t and comparing the outputs, which will be equal only if the the issuer used the public key for both executions. We also observe that these checks can be batched for an arbitrary number of issued tokens by computing a random linear combination of the values $\mathsf{H}_t(t_i)$, obtaining a VOPRF evaluation on that value, and comparing with the same linear combination of the other tokens. Thus a user can verify n tokens by running one additional token request only. We note further that removing the zero knowledge argument significantly simplifies the issuer work, which now consist only of one multiplication.

Applying the above idea to the anonymous token construction with private metadata bit is more challenging since the user does not know which of the two public keys the issuer will use. However, the client can unblind the response from the issuer using each of the public keys and thus obtain one valid and one random token. This property turns out to be true if the issuer behaves honestly but if the issuer is malicious, he can create public keys and a response W' such that the two values obtained from the unblinding with each of the public keys are correlated and this correlation can be used for fingerprinting the user. Thus, in our construction the user computes two values $T'_d = r_d(\mathsf{H}_t(t) - \rho_d G)$, $d \in \{0, 1\}$, the issuer uses one of them to compute his response $W' = x_b T'_b + y_b S'_b$ with a private bit b. The user unblinds W' using both public keys and the scalars r_d, ρ_d for $d \in \{0, 1\}$ to obtain S_0, W_0, S_1, W_1, which she uses for the final token. The resulting token verifies with only one of the issuer's keys: the key corresponding to the private metadata value.

Verification Oracle. 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 his choice. This is not explicitly supported in the current Privacy Pass security proof [22]. 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 was defined by Boneh et al. [9]. 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 an 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 that reads the private metadata bit of a token.*

Efficiency of Our Constructions. We consider the most expensive computation operation in the above schemes, which is elliptic curve multiplication, and the largest communication overhead, which is the number of group elements transferred. We report in Table 1 the efficiency of our constructions. Additionally, the variant of the constructions that supports validity verification oracle in the PMB security game adds the overhead of Okamoto–Schnorr Privacy Pass to the overhead of PMBTokens. The modifications of the constructions that do not use DLEQ or DLEQOR proofs save work for the issuer with no or moderate increase in communication and increased user computation. This computation trade-off is beneficial for settings where the issuer handles orders of magnitude more token

Table 1. Computational and Communication costs of our constructions, with and without zero-knowledge proofs.

Constructions	# multiplications		Communication
	user	issuer	(# elements)
Construction 1 (Privacy Pass) [22]	6	3	2
Construction 2 (Okamoto–Schnorr Privacy Pass)	9	6	2
Construction 3 (PMBTokens)	15	12	2
Construction 4 (Privacy Pass no DLEQ)	4	1	2
Construction 5 (PMBTokens no DLEQOR)	12	2	3

issuance requests than any particular user. We further implement our constructions in Rust, and report their practicals costs in Section 8. Using a Ristretto group on Curve25519, PMBTokens issuance runs in 822 µs and redemption takes 241 µs, while Privacy Pass issuance runs in 297 µs and redemption takes 95 µs. Without the discrete logarithm proofs, Construction 5 issuance runs in 144 µs and redemption takes 249 µs. Henceforth, PMBTokens and Construction 5 introduce a small overhead over Privacy Pass.

Paper Organization. We overview the hardness assumptions and the building block primitives we use in Section 2 and Appendices A and B. Section 3 defines our new anonymous tokens primitive and its security notions. We recall the Privacy Pass construction in Section 4, and present a (randomized) Okamoto–Schnorr anonymous tokens construction in Section 5. Next, Section 6 presents our construction for anonymous tokens with private metadata bit, called PMBTokens. Section 7 proposes modifications of Privacy Pass and PMBTokens that avoid the need of zero-knowledge proofs. Finally, Section 8 reports on the efficiency costs of our implementation. Due to space constraints we present most of our security proofs in Appendices C to G, and Appendix H describes a construction that supports verification oracle functionality.

1.2 Related Work

Starting with the work of Chaum [14], the concept of blind signatures has been widely used as a tool for building anonymous credentials. Blind Schnorr and Okamoto–Schnorr signatures, which have been studied and analyzed in the random oracle model [17, 39, 41, 43, 44, 27], require three moves of interaction between the user and the issuer. Blind signatures constructions that achieve one round, which is the goal for our construction, rely on more expensive building blocks.

The works of Boldyareva [8] and Bellare et al. [4] achieve round optimal (one round) constructions in the random oracle model under interactive assumptions. These constructions use the same blinding idea as the VOPRF [32] used by Privacy Pass, but are defined over groups where DDH is easy and CDH holds (or where the RSA assumptions hold), which enables public verifiability but requires larger group parameters. Other blind signature constructions have evolved from constructions that need a CRS [24, 45, 6] to constructions in the standard model [28, 26, 25], but they rely on bilinear groups with a pairing operation. This adds complexity to the group instantiations for schemes and computational cost, which we aim to minimize.

Partially blind signatures, for which we also have round-optimal constructions [24, 45, 7], allow the issuer to embed some information in the signature, however, this information is *public*, unlike the private metadata bit that is the goal of our construction.

Group signatures [18, 11, 3] present functionality which allows all member of the group to sign messages, with the property that signatures from different signers are indistinguishable. At the same time there is a master secret key that belongs to a group manager, which can be used to identify the signer of a message. We can view different signer keys as signing keys for the private metadata bits, and the master secret key as a way to read that bit value. Group blind signatures [38], which provide also the oblivious evaluation for the signing algorithm we aim at, provide a solution for the anonymous token functionality with a private metadata bit. Existing blind group signatures constructions [38, 42, 29] require multiple rounds of interaction for the oblivious signing and communication of many group elements.

Abdalla et al. [2] introduced a notion of blind message authentication codes (MACs), a secret key analog to blind signatures. They showed that this notion can exist only assuming a commitment of the private key, and showed how to instantiate that primitive with Chaum's blind signatures [14]. Davidson et al. [22] construct a similar private key functionality for anonymous tokens using a VOPRF [32]; it is called Privacy Pass and is the basis of this work.

Everspaugh et al. [23] introduce the primitive of a partially oblivious PRF, which analogously to blind signatures allows the party with the secret key to determine part of the input for the PRF evaluation. However, this input needs to be public for verifiability. The presented partially blind PRF uses bilinear groups and pairings. Jarecki et al. [33] show how to obtain a threshold variant of the partially oblivious PRF.

The work of Tsang et al. [48] presents a construction for blacklistable anonymous credentials using bilinear maps, which enables the issuer to create a blacklist of identities and the user can only generate an authentication token if she is not blacklisted; hence the user does find out whether she has been blacklisted in this process.

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 a 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 $\mathsf{A}, \mathsf{B} \in \mathsf{PPT}$ 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 $[\![a, \bot]\!] \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$.

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 recall in Appendix A the classical discrete logarithm (DLOG), decisional Diffie–Hellman (DDH), and computational Diffie–Hellman (CDH) assumptions, and recall here the chosen-target Diffie–Hellman (CTDH) assumption.

Chosen-target Diffie–Hellman. The chosen-target Diffie–Hellman (CTDH) assumption [8, 31], for the group generator GrGen, states that for all ℓ and for all $\mathsf{A} \in \mathsf{PPT}$, A has negligible advantage in solving CDH on $\ell + 1$ target group elements, even when given access to a CDH helper oracle for ℓ instances. More formally, for all ℓ and for all $\mathsf{A} \in \mathsf{PPT}$,

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

where $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$ is defined in Fig. 1. Note that for $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},0}(\lambda)$ is equivalent to CDH.

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

```
Oracle $\textsc{Target}(t_i)$
if $\exists (t_i, Y_i)$ in $T$
  return $Y_i$
else
  $Y_i \leftarrow\!\!\$\ \mathbb{G}$
  append $(t_i, Y_i)$ to $T$
  return $Y_i$
```

```
Oracle $\textsc{Help}(Y)$
$q := q + 1$
return $xY$
```

Fig. 1. The Chosen-Target Diffie–Hellman security game.

2.2 Non-interactive arguments of knowledge

A non-interactive zero-knowledge (NIZK) argument of knowledge $\mathsf{\Pi}$ for a relation R consists of the following three algorithms:

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

$\mathsf{\Pi}$ must satisfy the following three properties:

Completeness: A proof system $\mathsf{\Pi}$ is (perfectly) *complete* if for any $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, $\sigma \in [\mathsf{\Pi.Setup}(\Gamma)]$ and $(\phi, w) \in \mathsf{R}$:

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

Knowledge soundness: A proof system $\mathsf{\Pi}$ is *knowledge-sound* for R if for any PPT adversary A there exists a PPT extractor Ext such that:

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

and $\mathrm{KSND}_{\mathsf{\Pi},\mathsf{R},\mathsf{A},\mathsf{Ext}}(\lambda)$ is defined in Fig. 2 and $\mathsf{A.rl}(\lambda)$ is the randomness length for the machine A. An *argument of knowledge* is a knowledge-sound proof system.

Remark. In our proofs for ease of notation we will omit to specify explicitly that the extractor takes as input the coins of the adversary.

Zero Knowledge: A proof system $\mathsf{\Pi}$ is zero-knowledge if for any PPT adversary A:

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

```
Game KSND_{Π,R,A,Ext}(λ)
  $\Gamma := \mathsf{Setup}(1^\lambda); (\sigma,\tau) \leftarrow \Pi.\mathsf{Setup}(\Gamma)$
  $r \leftarrow_\$ \{0,1\}^{\mathsf{A.rl}(\lambda)}; (\phi,\pi) := \mathsf{A}(\sigma; r)$
  $w \leftarrow \mathsf{Ext}(\sigma, r)$
  return $(\Pi.\mathsf{V}(\sigma,\phi,\pi)$ and $\mathsf{R}(\phi,w) = \mathbf{false})$

Game $\mathrm{ZK}^{\beta}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)$
  $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
  $(\sigma,\tau) \leftarrow \Pi.\mathsf{Setup}(\Gamma)$
  $\beta' \leftarrow \mathsf{A}^{\mathrm{PROVE}_\beta}(\sigma)$
  return $\beta'$

Oracle $\mathrm{PROVE}_\beta(\phi, w)$
  if $(\phi, w) \notin \mathsf{R}$ then
    return $\perp$
  $\pi_0 \leftarrow \Pi.\mathsf{P}(\sigma,\phi,w)$
  $\pi_1 \leftarrow \Pi.\mathsf{Sim}(\sigma,\tau,\phi)$
  return $\pi_\beta$
```

Fig. 2. Games for knowledge soundness (KSND), and zero knowledge (ZK).

where $\mathrm{ZK}^{\beta}_{\Pi,\mathsf{R},\mathsf{A}}(\lambda)$ is defined in Fig. 2.

In our constructions we will use Sigma proof protocols made non-interactive using the Fiat-Shamir transform in the random oracle model. More specifically we will be using proof systems for the following languages (cf. Appendix B for constructions).

Proof systems for discrete logarithms knowledge (DLOG) with one and two generators, defined by the following languages

$$\mathcal{L}_{DLOG1} := \left\{ \begin{array}{c} X \in \mathbb{G} \ : \\ \text{⅄} x \in \mathbb{Z}_p \ , \ X = xG \end{array} \right\}; \ \mathcal{L}_{DLOG2} := \left\{ \begin{array}{c} X \in \mathbb{G} \ : \text{⅄} x, y \in \mathbb{Z}_p \\ X = xG + yH \end{array} \right\};$$

A proof system for discrete logarithms equality (DLEQ) with one generator defined by the following language

$$\mathcal{L}_{DLEQ1} := \left\{ (X,T,W) \in \mathbb{G}^3 \ : \ \text{⅄} x \in \mathbb{Z}_p \ , \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} \right\};$$

A proof system for discrete logarithms equality (DLEQ) with two generators defined by the following language

$$\mathcal{L}_{DLEQ2} := \left\{ (X,T,S,W) \in \mathbb{G}^4 \ : \ \text{⅄}(x,y) \in \mathbb{Z}_p^2 \ , \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\};$$

A proof system for OR of discrete logarithms equalities (DLEQOR) with two generators defined by the following language

$$\mathcal{L}_{DLEQOR2} := \left\{ \begin{array}{c} (\mathbf{X} := (X_0, X_1), T, S, W) \in \mathbb{G}^5 \ : \\ \text{⅄}(b,x,y) \in [2] \times \mathbb{Z}_p \times \mathbb{Z}_p \ , \ \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \end{array} \right\}.$$

3 Anonymous Tokens

In this section, we describe the functionality and the security properties of the anonymous tokens primitive we construct.

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 user can later use the token as a trust signal. 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 private metadata bit is hidden, but can be recovered by the issuer when the token is redeemed.

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:*

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$ – *a key generation algorithm that generates a private key* sk *along with some set of public parameters* pp*;*
- $\mathsf{token} \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b)\rangle$ – *a signing protocol that involves the interactive algorithms* $\mathsf{AT.Usr}$ *(run by the user) with input value* $t \in \{0,1\}^\lambda$ *and* $\mathsf{AT.Sig}$ *(run by the issuing server) with input its private key and a bit* b *. At the end of the interaction, the server outputs nothing while the user outputs a anonymous token* $\mathsf{token} := (t, \sigma)$ *if it terminates correctly and* $\mathsf{token} := \perp$ *otherwise. For a* 1*-round protocol, the interaction can be realized by the following algorithms:*

$$\begin{aligned} (msg_U, \mathsf{st}_U) &\leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t) \\ msg_S &\leftarrow \mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, msg_U) \\ \mathsf{token} = (t, \sigma) &\leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_U, msg_S) \end{aligned}$$

- *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 an invalid state.

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

$$\Pr[\mathsf{AT.VerValid}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b)\rangle) = 1] = 1 - \mathsf{negl}(\lambda)\,,$$
$$\Pr[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b)\rangle) = b] = 1 - \mathsf{negl}(\lambda).$$

3.2 Security Properties for Anonymous Tokens

We proceed to define the security properties relevant for the anonymous tokens (with 1-round signing protocols).

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. 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 *with 1-round signing protocol is* one-more unforgeable*, if for all* $\mathsf{A} \in \mathsf{PPT}$*, for all* ℓ*,*

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

where $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ *is defined in Fig. 3, and where* A *can invoke the oracle* $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$ ℓ *times for each private metadata bit value, the validity verification oracle* $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot)$*, and the bit reading oracle* $\mathsf{AT.ReadBit}(\mathsf{sk}, \cdot)$.

Unlinkability. This 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 precisely, if the issuer has yet to redeem m tokens, and is presented a token, we limit the probability that it can guess from which execution this token was coming from, even if it sees all the $m - 1$ remaining tokens in a permuted order.

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

$(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$
$(t_i, \sigma_i)_{i=1}^{\ell+1} \leftarrow \mathsf{A}^{\mathsf{AT.Srv}_1(\mathsf{p},\mathsf{sk},\cdot,\cdot),\ \mathsf{AT.VerValid}(\mathsf{sk},\cdot),\ \mathsf{AT.ReadBit}(\mathsf{sk},\cdot)}(\mathsf{pp})$
return $t_i$ all different and $\mathsf{AT.VerValid}(\mathsf{sk}, (t_i, \sigma_i)) = 1\ \forall i \in [1, \ell+1]$
        and $\exists b' \in \{0,1\}$ such that $\mathsf{AT.ReadBit}(\mathsf{sk}, (t_i, \sigma_i)) = b'\ \forall i \in [1, \ell+1]$
```

Fig. 3. One-more unforgeability game for an anonymous token scheme AT with a 1-round signing protocol.

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

$\mathit{\Gamma} \leftarrow \mathsf{GrGen}(1^\lambda)$;   $(\mathsf{st}, \mathsf{pp}) \leftarrow \mathsf{A}(\mathit{\Gamma})$
$k_1 := 0; k_2 := 0; U := \emptyset$
$(\mathsf{st}, \{msg_i\}_{i \in U}) \leftarrow \mathsf{A}^{\mathrm{Usr}_0(),\mathrm{Usr}_1(\cdot,\cdot)}(\mathsf{st})$
if $U = \emptyset$ then return $0$
// compute a challenge token
$j \leftarrow_\$ U$;   $U := U \setminus \{j\}$
$\mathsf{token}_j := \mathsf{AT.Usr}_1(\mathsf{st}_j, msg_j)$
// compute and permute other tokens
$\forall i \in U, \mathsf{token}_i := \mathsf{AT.Usr}_1(\mathsf{st}_i, msg_i)$
$\phi \leftarrow \mathcal{S}_U$
$j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \{\mathsf{token}_{\phi(i)}\}_{i \in U})$
return $k_1 - k_2 \geq m$ and $j' = j$
```

```
Oracle $\mathrm{Usr}_0()$

$k_1 := k_1 + 1$// session id
$t_{k_1} \leftarrow_\$ \{0,1\}^\lambda$
$(T'_{k_1}, \mathsf{st}_{k_1}) \leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t_{k_1})$
$U := U \cup \{k_1\}$// open sessions
return $(k_1, T'_{k_1})$
```

```
Oracle $\mathrm{Usr}_1(j, msg)$

if $j \notin U$ then return $\perp$
$\mathsf{token} \leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_j, msg)$
if $\mathsf{token} \neq \perp$ then
    $U := U \setminus \{j\}$;   $k_2 := k_2 + 1$
return $\mathsf{token}$
```

Fig. 4. Unlinkeability game for an anonymous token scheme AT with a 1-round signing protocol. For a set X, $\mathcal{S}_X$ denotes the symmetric group of X, i.e., the group of all permutations of X.

Definition 3 (Unlinkability). *An anonymous token scheme* AT *with* 1*-round signing protocol is* κ-unlinkable*, if for all adversaries* $\mathsf{A} \in \mathsf{PPT}$ *and for all* $m \in \mathbb{Z}, m > 0$*:*

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

where $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)$ *is defined in Fig. 4, and where* A *can invoke the oracle* $\mathrm{Usr}_1(\cdot,\cdot)$ *at most* m *times less than the number of times it invokes* $\mathrm{Usr}_0()$*.*

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

```
Game $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$
$(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^{\lambda})$
$\beta' \leftarrow \mathsf{A}^{\mathsf{AT.Srv}_1(\mathsf{pp},\mathsf{sk},\cdot,\cdot),\mathsf{AT.Srv}_1(\mathsf{pp},\mathsf{sk},\beta,\cdot),\mathsf{AT.VerValid}(\mathsf{sk},\cdot)}(\mathsf{pp})$
return $\beta'$
```

Fig. 5. Private metadata bit game for an anonymous token scheme AT with a 1-round signing protocol.

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.

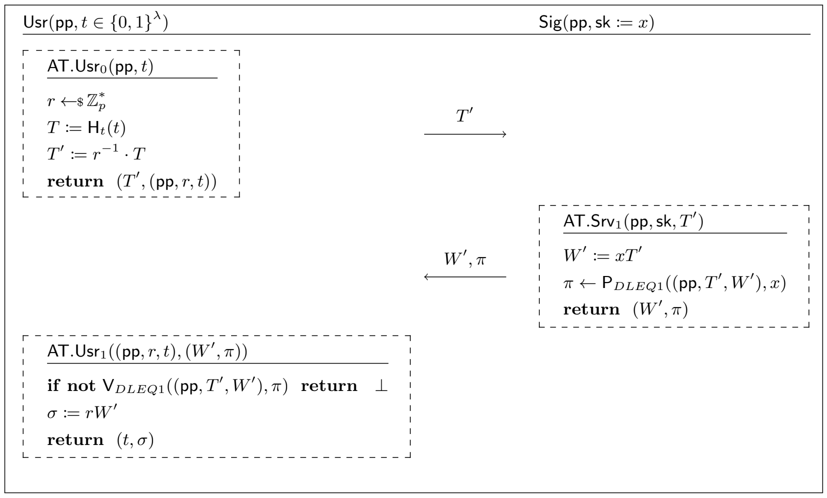
Definition 4 (Private Metadata Bit). *An anonymous token scheme* AT *with* 1*-round signing protocol provides* private metadata bit *if for all adversary* $A \in \mathsf{PPT}$ *the advantage:*

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

where $\mathrm{PMB}^{\beta}_{\mathsf{AT},A}(\lambda)$ *is defined in Fig. 5, and where* A *has access to the following oracles: a signing oracle* $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$ *where* A *can provide both the message and the bit to be used for the token, a signing oracle* $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \beta, \cdot)$ *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 the notation from Section 3, the anonymous token protocol proposed in [22] (under the name Privacy Pass), and built on top of the VOPRF as described in [32]. The Privacy Pass construction uses a Schnorr-style DLEQ proof for the verifiability in the issuance phase. Note that this anonymous token protocol is deterministic, i.e., there will exist a unique value $\sigma \in \mathbb{G}$ corresponding to a string $t \in \{0, 1\}^{\lambda}$ 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 to enable randomized tokens (Section 5) and we will eventually extend the construction to support private metadata bit (Section 6 and Appendix H).



```
Usr(pp, $t \in \{0,1\}^\lambda$)                                   Sig(pp, sk $:= x$)

AT.Usr$_0$(pp, $t$)
    $r \leftarrow_\$ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r^{-1} \cdot T$
    return $(T', (\mathsf{pp}, r, t))$
                                  $\xrightarrow{T'}$
                                                                   AT.Srv$_1$(pp, sk, $T'$)
                                                                       $W' := xT'$
                                  $\xleftarrow{W', \pi}$                    $\pi \leftarrow \mathsf{P}_{DLEQ1}((\mathsf{pp}, T', W'), x)$
                                                                       return $(W', \pi)$
AT.Usr$_1$((pp, $r$, $t$), $(W', \pi)$)
    if not $\mathsf{V}_{DLEQ1}((\mathsf{pp}, T', W'), \pi)$ return $\perp$
    $\sigma := rW'$
    return $(t, \sigma)$
```

Fig. 6. Token issuance for Construction 1 (Privacy Pass).

Construction 1 (Privacy Pass) *Let* $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be an algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs a random generator* G*. Let* $(\mathsf{P}_{DLEQ1}, \mathsf{V}_{DLEQ1})$ *be a proof system for the DLEQ relationship defining the language* $\mathcal{L}_{DLEQ1}$*.*

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

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run* $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ *to obtain group parameters.* Γ *will be an implicit input to all other algorithms.*
  - *Sample a random invertible value* $x \leftarrow_\$ \mathbb{Z}_p^*$*, and set* $\mathsf{sk} := x$*,* $\mathsf{pp} := xG$*.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}) \rangle$ – *the anonymous token issuance protocol is defined in Fig. 6.*
- *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. Therefore, the client always returns a tuple $(t, \sigma) \in \{0,1\}^\lambda \times \mathbb{G}$ such that

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

Security. We prove unforgeability and 1-unlinkability in Appendix C.

5 Okamoto–Schnorr Privacy Pass

In this section, we describe a novel anonymous token protocol that builds on top of Privacy Pass (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 (Section 6 and Appendix H). While Privacy Pass uses a Schnorr-style DLEQ proof for the verifiability in the issuance phase, this new construction uses the corresponding Okamoto–Schnorr-style [39] variant of the DLEQ proof protocol. A different approach towards randomization of the deterministic evaluation algorithm could be leveraging pairings (cf. Section 1.2), but we do not pursue this approach here for efficiency reasons.

```
$\mathsf{Usr}(\mathsf{pp}, t \in \{0,1\}^\lambda)$                                   $\mathsf{Sig}(\mathsf{pp}, \mathsf{sk} \coloneqq (x, y))$

$\mathsf{AT.Usr}_0(\mathsf{pp}, t)$
  $r \leftarrow_{\$} \mathbb{Z}_p^*$
  $T \coloneqq \mathsf{H}_t(t)$
  $T' \coloneqq r^{-1} \cdot T$
  return $(T', (\mathsf{pp}, r, t, T'))$

                          $\xrightarrow{T'}$

                                        $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk} = (x, y), T')$
                                          $s \leftarrow_{\$} \{0,1\}^\lambda$
                                          $S' \coloneqq \mathsf{H}_s(T', s)$
                                          $W' \coloneqq xT' + yS'$
                                          $\pi \leftarrow \mathsf{P}_{DLEQ2}((\mathsf{pp}, T', S', W'), (x, y))$
                                          return $(s, W', \pi)$

                          $\xleftarrow{s, W', \pi}$

$\mathsf{AT.Usr}_1((\mathsf{pp}, r, t, T'), (s, W', \pi))$
  $S' \coloneqq \mathsf{H}_s(T', s)$
  if not $\mathsf{V}_{DLEQ2}((\mathsf{pp}, T', S', W'), \pi)$ then
    return $\perp$
  $S \coloneqq rS'$
  $W \coloneqq rW'$
  $\sigma \coloneqq (S, W)$
  return $(t, \sigma)$
```

Fig. 7. Token issuance for Construction 2 (Okamoto–Schnorr Privacy Pass).

Construction 2 (Okamoto–Schnorr Privacy Pass) *Let* $\Gamma \coloneqq (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs two distinct random generators* G *and* H *(in particular, the discrete log of*

H with respect to G is unknown). 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 $(\mathsf{P}_{DLEQ2}, \mathsf{V}_{DLEQ2})$ be a proof system for the DLEQ relationship defining the language $\mathcal{L}_{DLEQ2}$.

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

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run $\Gamma \coloneqq (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ to obtain group parameters. Γ will be an implicit input to all other algorithms.*
  - *Sample two random invertible values $x, y \leftarrow_\$ \mathbb{Z}_p^*$, and set $X \coloneqq xG + yH$.*
  - *Set $\mathsf{sk} \coloneqq (x, y)$ and $\mathsf{pp} \coloneqq X$.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Fig. 7.*
- *bool* $\leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$*:*
  - *Parse $\sigma = (S, W)$ and $\mathsf{sk} = (x, y)$.*
  - *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 Privacy Pass as defined in Section 4.

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

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

Unforgeability. We prove one-more unforgeability of the construction in Appendix D.

5.1 Unlinkability

In this section, we will prove that Construction 2 is 1-unlinkable (cf. Definition 3), which means that the probability that an adversary can guess which of m tokens not redeemed yet is upper-bounded by $1/m + \mathsf{negl}(\lambda)$.

Theorem 6. *Construction 2 provides 1-unlinkability (Definition 3) assuming the hardness of DDH and a knowledge sound proof system for the language $\mathcal{L}_{DLEQ2}$.*

Proof. The theorem trivially holds for $m = 1$. Let $m > 1$. We prove the theorem by a sequence of hybrids presented in Fig. 8.

Hyb_1 This hybrid is the execution of the unlinkability game, where the adversary is given as a challenge the j-th token.

Hyb_2 This hybrids unrolls $\mathsf{AT.Usr}_0$ in $\textsc{Usr}_0$. The distribution is unchanged.

Hyb_3 This hybrid returns 0 if U is empty after having removed j (i.e., if the set U was equal to $\{j\}$ in the first place). Note that, in that case, $|U| = k_1 - 1 - k_2$ and the winning condition $1 = k_1 - k_2 \geq m$ is not verified; hence the output is indistinguishable from the previous hybrid.

Hyb_4 This hybrid extracts another element k from U, and shuffles the rest. The distribution is the same as in the previous hybrid.

Hyb_5 This hybrid uses the knowledge extractor for the DLEQ proof to extract a witness (x, y) in $\mathsf{AT.Usr}_1$, and check that the witness verifies the relation. This hybrid is indistinguishable from the previous by the knowledge soundness of the proof system. Hence we have

$$k_1 \mathsf{Adv}^{\mathrm{ksnd}}_{DLEQ2,\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|,$$

where k_1 is the total number of calls to $\mathsf{AT.Usr}_1$.

Hyb_6 In this hybrid, instead of computing all the W's by unblinding the elements W''s provided by the adversary, we compute it ourselves using the (valid) extracted witnesses (x, y). The distribution is unchanged.

Hyb_7 This hybrid proceeds exactly as the previous one, except now all S's are sampled uniformly at random from $\mathbb{G}$.

If there exists $\mathsf{A} \in \mathsf{PPT}$ for which the outcome of the two hybrids is different, then it is possible to construct an adversary $\mathsf{B} \in \mathsf{PPT}$ 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 $(P, A := aP, B := bP, C) \in \mathbb{G}^4$ where $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$ and has to distinguish $C := abP$ (the case $\beta = 0$) from a uniformly distributed element over $\mathbb{G}$ (the case $\beta = 1$). B runs the game as per Hyb_6, except in $\mathsf{AT.Usr}_0$, instead of sampling T'_{k_1} uniformly at random from $\mathbb{G}$, B sets them as $T'_{k_1} = \gamma_{k_1} P$ where γ_{k_1} is random. Instead of computing $T_{k_1} = r_{k_1} T'_{k_1}$, B sets $T_{k_1} = \gamma_{k_1}\alpha_{k_1} A + \gamma_{k_1}\alpha'_{k_1} P = (\alpha_{k_1} a + \alpha'_{k_1})\gamma_{k_1} P = (\alpha_{k_1} a + \alpha'_{k_1}) T'_{k_1}$, i.e. implicitly setting $r_{k_1} := \alpha_{k_1} a + \alpha'_{k_1}$. For every query $\mathsf{H}_s(T'_{k_1}, s)$, B sample $\beta_{k_1,s}, \beta'_{k_1,s}$ at random and programs $\mathsf{H}_s(T'_{k_1}, s) = \gamma_{k_1}\beta_{k_1,s} B + \gamma_{k_1}\beta'_{k_1,s} P = (\beta_{k_1,s} b + \beta'_{k_1,s})\gamma_{k_1} P$. Finally, when computing S at index i, B sets $S_i := \alpha_i \beta_{i,s_i} \gamma_i C + \alpha_i \beta'_{i,s_i} \gamma_i A +$

$\alpha'_i \beta_{i,s_i} \gamma_i B + \alpha'_i \beta'_{i,s_i} \gamma_i P$. If C is random, then S_i is random and the distribution coincides with that of Hyb_7. If $C = abP$, then

$$\begin{aligned} S_i &= \alpha_i \beta_{i,s_i} \gamma_i (abP) + \alpha_i \beta'_{i,s_i} \gamma_i (aP) + \alpha'_i \beta_{i,s_i} \gamma_i (bP) + \alpha'_i \beta'_{i,s_i} \gamma_i P \\ &= \alpha_i \beta_{i,s_i} ab(\gamma_i P) + \alpha_i a \beta'_{i,s_i} (\gamma_i P) + \alpha'_i b \beta_{i,s_i} (\gamma_i P) + \alpha'_i \beta'_{i,s_i} (\gamma_i P) \\ &= (\alpha_i a + \alpha'_i)(\beta_{i,s_i} b + \beta'_{i,s_i})(\gamma_i P)\,, \end{aligned}$$

which results in the distribution of Hyb_6. It follows therefore that the advantage in distinguishing the two hybrids is, for all $\mathsf{A} \in \mathsf{PPT}$:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_7}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_6}_{\mathsf{AT},\mathsf{A}}(\lambda) \right| .$$

Hyb_8 This hybrid aborts if the adversary has already queried $\mathsf{H}_t(t_{k_1})$ before running $\mathsf{AT.Usr}_0$. This happens with negligible probability:

$$\frac{k_1}{2^\lambda} \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_8}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_7}_{\mathsf{AT},\mathsf{A}}(\lambda) \right| .$$

Hyb_9 In this hybrid, we program the random oracle so that T'_{k_1} is sampled at random, and we program the random oracle so that $T_{k_1} = \mathsf{H}_t(t_{k_1}) := r_{k_1} T'_{k_1}$. The distribution is unchanged.

Hyb_{10} This hybrid aborts if the extracted witnesses at positions j and k differ, and then sets $W_j := x_k \mathsf{H}_t(t_j) + y_k S_j$ and $W_k := x_j \mathsf{H}_t(t_k) + y_j S_k$ (which does not change the distributions).
If the game aborts at this step, by soundness of the proof system we would have that $\mathsf{pp} = x_j G + y_j H = x_k G + y_k H$ and thus $(y_k - y_j)/(x_j - x_k)$ is the discrete log of H base G. (if the two witnesses are different, it must be that $x_j \neq x_k$, and thus the inverse of $(x_j - x_k)$ exists.) It is therefore possible to construct an adversary $\mathsf{B} \in \mathsf{PPT}$ for $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$: the adversary B obtains a group description $\Gamma := (\mathbb{G}, p, \tilde{G})$ and a challenge $\tilde{H}$. It sets $G := \tilde{G}$ and $H := \tilde{H}$ and runs exactly as per Hyb_5. It extracts the two witnesses (x_j, y_j), and (x_k, y_k) and returns $(y_k - y_j)/(x_j - x_k)$. The adversary wins every time that Hyb_{10} aborts.

Hyb_{11} In the previous hybrid, r_j and r_k are only used in order to compute T_j and T_k. Therefore, they look completely random to the adversary, and so do S_j and S_k. In this hybrid, we can therefore swap the indices j and k without the adversary noticing.

We obtain that

$$\begin{aligned} \mathsf{Adv}^{\mathsf{Hyb}_{11}}_{\mathsf{AT},\mathsf{A}}(\lambda) &\leq \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + k_1 \mathsf{Adv}^{\mathrm{ksnd}}_{\mathsf{DLEQ},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \frac{k_1}{2^\lambda} \\ &= \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) + \mathsf{negl}(\lambda)\,. \end{aligned}$$

```
Game UNLINK$_{\mathsf{AT},\mathsf{A},m}(\lambda)$ in Hyb$_1$, Hyb$_3$ , Hyb$_4$ , $\boxed{\text{Hyb}_{10}}$
$\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{st}, \mathsf{pp}) \leftarrow \mathsf{A}(\Gamma)$
$k_1 := 0; k_2 := 0; U := \emptyset$
$(\mathsf{st}, \{(s_i, W_i', \pi_i)\}_{i \in U}) \leftarrow \mathsf{A}^{\textsc{Usr}_0(), \textsc{Usr}_1(\cdot,\cdot)}(\mathsf{st})$
if $U = \emptyset$ then return 0
$j \leftarrow\$ U$;  $U := U \setminus \{j\}$
$\mathsf{token}_j := \mathsf{AT.Usr}_1(\mathsf{st}_j, (s_j, W_j', \pi_j))$
if $U = \emptyset$ then return 0
$k \leftarrow\$ U$;  $U := U \setminus \{k\}$
$\mathsf{token}_k := \mathsf{AT.Usr}_1(\mathsf{st}_k, (s_k, W_k', \pi_k))$
$\forall i \in U, \mathsf{token}_i := \mathsf{AT.Usr}_1(\mathsf{st}_i, (s_i, W_i', \pi_i))$
$\boxed{\textbf{if not } (x_i, y_i) = (x_k, y_k) \textbf{ then abort}}$
$\boxed{W_j := x_k \mathsf{H}_t(t_j) + y_k S_j}$
$\boxed{\textbf{if } \mathsf{V}_{DLEQ2}((\mathsf{pp}, T_j', S_j', W_j'), \pi_j) \textbf{ then } \mathsf{token}_j := (t_j, (S_j, W_j))}$
$\boxed{W_k := x_j \mathsf{H}_t(t_k) + y_j S_k}$
$\boxed{\textbf{if } \mathsf{V}_{DLEQ2}((\mathsf{pp}, T_k', S_k', W_k'), \pi_k) \textbf{ then } \mathsf{token}_k := (t_k, (S_k, W_k))}$
$\phi \leftarrow \mathcal{S}_U$
$j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \{\mathsf{token}_{\phi(i)}\}_{i \in U})$
$j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \mathsf{token}_k, \{\mathsf{token}_{\phi(i)}\}_{i \in U})$
return $k_1 - k_2 \geq m$ and $j' = j$
```

```
Oracle Usr$_1(i, msg = (s_i, W_i', \pi_i'))$ in Hyb$_1$
if $i \notin U$ then return $\perp$
$\mathsf{token}_i \leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_i, msg)$
if $\mathsf{token}_i \neq \perp$ then
    $U := U \setminus \{i\}$
    $k_2 := k_2 + 1$
return $\mathsf{token}_i$
```

```
Oracle Usr$_0()$ in Hyb$_2$ , Hyb$_8$ , $\boxed{\text{Hyb}_9}$
$k_1 := k_1 + 1$
$t_{k_1} \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_t(t_{k_1})$ was queried then abort
$(T_{k_1}', \mathsf{st}_{k_1}) \leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t_{k_1})$
$r_{k_1} \leftarrow\$ \mathbb{Z}_p^*$
$T_{k_1} := \mathsf{H}_t(t_{k_1})$
$T_{k_1}' := r_{k_1}^{-1} \cdot T_{k_1}$
$\boxed{T_{k_1}' \leftarrow\$ \mathbb{G}}$
$\boxed{T_{k_1} := \mathsf{H}_t(t_{k_1}) := r_{k_1} T_{k_1}'}$
$\mathsf{st}_{k_1} := (\mathsf{pp}, r_{k_1}, t_{k_1}, T_{k_1}')$
$U := U \cup \{k_1\}$
return $(k_1, T_{k_1}')$
```

```
$\mathsf{AT.Usr}_1(\mathsf{st}, (s, W', \pi))$ in Hyb$_1$, Hyb$_5$ , Hyb$_6$ , $\boxed{\text{Hyb}_7}$
$(\mathsf{pp}, r, t, T') := \mathsf{st}$
$S' := \mathsf{H}_s(T', s)$
if not $\mathsf{V}_{DLEQ2}((\mathsf{pp}, T', S', W'), \pi)$ then return $\perp$
$S := rS'$;  $\boxed{S \leftarrow\$ \mathbb{G}}$
$(x, y) \leftarrow \mathsf{Ext}((X, T', S', W'), \pi)$
if not $R((x, y), (X, T', S', W'))$ then abort
$W := rW'$;  $W := x\mathsf{H}_t(t) + yS$
$\sigma := (S, W)$
return $(t, \sigma)$
```

Fig. 8. Summary of hybrid changes for proof of unlinkability of Construction 2. We recall that $\mathcal{S}_X$ denotes the symmetric group of X, i.e., the group of all permutations of X.

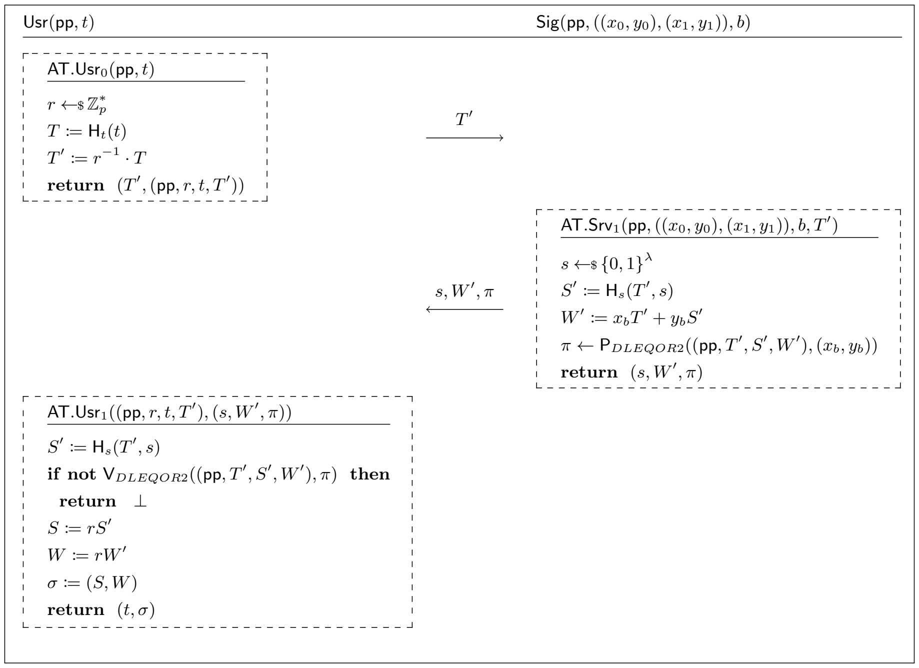
Now, it is sufficient to bound the probability of success adversary in Hyb$_{11}$. Since the index that the adversary needs to guess j is independent of the challenge it receives, we conclude that

$$\Pr\left[\text{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) = 1\right] \leq \frac{1}{k_1 - k_2} + \mathsf{negl}(\lambda) \leq \frac{1}{m} + \mathsf{negl}(\lambda)\,.$$

which completes the proof. □

6 Private Metadata Bit Tokens

In this section, we present PMBTokens, an extension of the anonymous token construction from Section 5 that supports a private metadata bit,



```
Usr(pp, $t$)                                        Sig(pp, $((x_0, y_0), (x_1, y_1)), b$)

AT.Usr$_0$(pp, $t$)
    $r \leftarrow\$ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r^{-1} \cdot T$
    return $(T', (\mathsf{pp}, r, t, T'))$

                        $\xrightarrow{T'}$

                                                    AT.Srv$_1$(pp, $((x_0, y_0), (x_1, y_1)), b, T'$)
                                                        $s \leftarrow\$ \{0,1\}^\lambda$
                                                        $S' := \mathsf{H}_s(T', s)$
                                                        $W' := x_b T' + y_b S'$
                                                        $\pi \leftarrow \mathsf{P}_{DLEQOR2}((\mathsf{pp}, T', S', W'), (x_b, y_b))$
                                                        return $(s, W', \pi)$

                        $\xleftarrow{s, W', \pi}$

AT.Usr$_1$($(\mathsf{pp}, r, t, T'), (s, W', \pi)$)
    $S' := \mathsf{H}_s(T', s)$
    if not $\mathsf{V}_{DLEQOR2}((\mathsf{pp}, T', S', W'), \pi)$ then
        return $\perp$
    $S := rS'$
    $W := rW'$
    $\sigma := (S, W)$
    return $(t, \sigma)$
```

Fig. 9. Token issuance for Construction 3 (PMBTokens).

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 we will "commit" to both secret keys (by creating public 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 public keys has been used.

Note that our construction generalizes directly to more than two values for the private metadata, since using more public keys and a DLEQOR argument of knowledge for statement with more clauses. However, for clarity of presentation, we focus on the setting of one private bit, and also since this comes with a trade-off for the unlinkability property.

Remark 7. 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 Appendix H that will provide a validity verification.

Construction 3 (PMBTokens) *Let $\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be algorithm that generates a group $\mathbb{G}$ of order p and outputs two distinct random generators G and H (in particular, the discrete log of H with respect to G is unknown). 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 $(\mathsf{P}_{DLEQ2}, \mathsf{V}_{DLEQ2})$ be a proof system for the DLEQ relationship defining the language $\mathcal{L}_{DLEQ2}$.*

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

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run $\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ to obtain group parameters. Γ will be an implicit input to all other algorithms.*
  - *Sample four random invertible values $x_0, x_1, y_0, y_1 \leftarrow_\$ \mathbb{Z}_p^*$, and set $X_0 := x_0 G + y_0 H$ and $X_1 := x_1 G + y_1 H$. Restart if $X_0 = X_1$.*
  - *Set $\mathsf{sk} := ((x_0, y_0), (x_1, y_1))$ and $\mathsf{pp} := (X_0, X_1)$.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b)\rangle$ *– the anonymous token issuance protocol is defined in Fig. 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 $W = x_0 \mathsf{H}_t(t) + y_0 S$ and $W \neq x_1 \mathsf{H}_t(t) + y_1 S$, return* 0
  - *If $W \neq x_0 \mathsf{H}_t(t) + y_0 S$ and $W = x_1 \mathsf{H}_t(t) + y_1 S$, 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 $W = x_0 \mathsf{H}_t(t) + y_0 S = x_1 \mathsf{H}_t(t) + y_1 S$ and $y_0 \neq y_1$ (if $y_0 = y_1$, the equation holds only if $x_0 = x_1$, which is impossible by construction), it means that:

$$\mathsf{H}_t(t) = \frac{x_0 - x_1}{y_0 - y_1} S.$$

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, for all $b \in \{0,1\}$,

$$\Pr\left[\mathsf{AT.ReadBit}(\mathsf{sk}, \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b)\rangle) = b\right] = 1 - \mathsf{negl}(\lambda)\,.$$

6.1 Security.

In Appendix E, we prove the one-more unforgeability and **2**-unlinkability of the construction, which means that the probability that an adversary can guess which of m tokens not redeemed yet is upper-bounded by $2/m + \mathsf{negl}(\lambda)$. Indeed, they key idea is that the adversary can now embed different private metadata bits during the issuances, halving its search space at most.

6.2 Privacy of the Metadata Bit

Theorem 8. *Construction 3 provides privacy for the metadata bit (Definition 4) assuming the hardness of DDH and a zero-knowledge proof system for the language* $\mathcal{L}_{DLEQOR2}$.

Proof. We consider the sequence of hybrids presented in Fig. 21 that transitions from an execution 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). 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 public parameters $\mathsf{pp} \coloneqq (X_0, X_1)$. 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 This hybrid replaces the way zero-knowledge proofs are generated in $\mathsf{AT.Srv}_1$: instead of using the proving algorithm $\mathsf{P}_{DLEQOR2}$, we use the zero-knowledge simulator $\mathsf{Sim}_{DLEQOR2}$.
If there exists an adversary $\mathsf{A} \in \mathsf{PPT}$ whose output is different between the two games, then it is possible to construct an adversary for

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b}, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\pi \leftarrow \mathsf{P}_{DLEQOR2}((X_0, X_1, T', S', W'), (x_{\hat{b}}, y_{\hat{b}}))$
$\pi \leftarrow \mathsf{Sim}_{DLEQOR2}(X_0, X_1, T', S', W')$
return $(s, W', \pi)$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_2(\lambda)$, $\mathsf{Hyb}_3(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$\pi \leftarrow \mathsf{P}_{DLEQOR2}((X_0, X_1, T', S', W'), (x_b, y_b))$
$\pi \leftarrow \mathsf{Sim}_{DLEQOR2}(X_0, X_1, T', S', W')$
return $(s, W', \pi)$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b}, T')$ in $\mathsf{Hyb}_4(\lambda), \mathsf{Hyb}_5(\lambda), \mathsf{Hyb}_6(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then) : abort
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat{b}} T' + y_{\hat{b}} S'$
$\pi \leftarrow \mathsf{Sim}_{DLEQOR2}(X_0, X_1, T', S', W')$
return $(s, W', \pi)$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$ in $\mathsf{Hyb}_4(\lambda)$, $\mathsf{Hyb}_5(\lambda)$, $\boxed{\mathsf{Hyb}_6(\lambda)}$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$y' \leftarrow\$ \mathbb{Z}_p^*$;   $W' := x_b T' + y' S'$
$W' := x_{1-b} T' + y' S'$
$\boxed{W' := x_{1-b} T' + y_{1-b} S'}$
$\pi \leftarrow \mathsf{Sim}_{DLEQOR2}(X_0, X_1, T', S', W')$
return $(s, W', \pi)$
```

Fig. 10. Summary of the proof for privacy of the metadata bit of Construction 3.

the underlying zero-knowledge of the proof system: consider the adversary $\mathsf{B} \in \mathsf{PPT}$ for the game $\mathrm{ZK}^{\beta}_{DLEQOR2}(\lambda)$ 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$ (and a $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$) query, after generating the values s, S', and W' as per Hyb_0, the proof π is generated via the $\textsc{Prove}_\beta$ oracle for the statement $((X_0, X_1), T', S', W') \in \mathcal{L}_{DLEQOR2}$. At the end of its execution, A (and so B) return a guess b'.
If the $\textsc{Prove}_\beta$ oracle outputs proofs via $\mathsf{P}_{DLEQOR2}$, the game is identical to Hyb_0, else the game is identical to Hyb_1. It follows that, for any adversary $\mathsf{A} \in \mathsf{PPT}$, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in DLEQOR, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQOR2},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{AT},\mathsf{A}}(\lambda).\right|$$

Hyb_2 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 $\mathsf{A} \in \mathsf{PPT}$ making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles

H_s, $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$, or $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$, the probability that the game aborts is at most $q(q-1)/2p$. It follows that:

$$q(q-1)/(2p) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

Hyb_3 We 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$ 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_2, 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$, we sample $s \leftarrow\!\!\$\, \{0,1\}^\lambda$ and check for collisions w.r.t. previous queries to H_s as per Hyb_2. 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b} = b, T')$, 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b} = 1 - b, T')$ is handled exactly as per Hyb_2.

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_2; if the challenge C is provided according to $\mathrm{DDH}^1_{\mathsf{A},\mathsf{GrGen}}(\lambda)$, B behaves exactly as per Hyb_3. 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 $\mathsf{Sim}_{DLEQOR2}$ fails), B will distinguish a random tuple from a DDH tuple. It follows therefore that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

Hyb_4 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_3 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_5 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 $\mathsf{Hyb}_2 \to \mathsf{Hyb}_3$. Therefore, it follows that:

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{B},\mathsf{GrGen}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|$$

At this point we note that the oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$ 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_2 (via the same argument used for the transition $\mathsf{Hyb}_1 \to \mathsf{Hyb}_2$), and swap back the zero-knowledge simulator with the prover's algorithm $\mathsf{P}_{DLEQOR2}$ (via the same argument used for the transition $\mathsf{Hyb}_0 \to \mathsf{Hyb}_1$). Therefore, the advantage of an adversary A in winning the game $\mathrm{PMB}^\beta_{\mathsf{AT},\mathsf{A}}(\lambda)$

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

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 $\lambda = \lfloor \log_2 p \rfloor$. $\square$

6.3 Enabling Validity Verification in the PMB Game

PMBTokens (Construction 3) does not support validity verification functionality, which could be accessible by the adversary for the private metadata security game. We can enable such functionality by combining it with the Okamoto–Schnorr Privacy Pass token functionality into one token that has two parts: a token that has no private metadata and can be used for validity verification, and a second token, which provides a private metadata bit. It is important that these two parts could not be separated (they will depend on the same $\mathsf{H}_t(t), S$ values) and used independently for the purpose of reading the metadata bit. We present in detail in Appendix H the design combining Constructions 2 and 3.

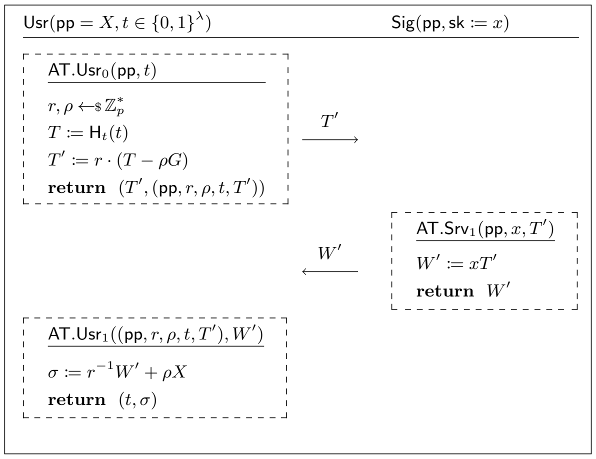
Jumping ahead, we can also instantiate this design by combining Constructions 4 and 5 that do not use ZK proofs. In the later case the unlinkability will degrade to 6-unlinkability since the issuer can cause each of the two token to be invalid independently.

7 Anonymous Tokens Without ZK

In this section, we present modifications of both Privacy Pass and PMBTokens constructions that avoids the DLEQ and the DLEQOR proofs. We recall that these proofs provide verifiability for the user that allows her to check that the tokens she has received are consistent with the issuer's public key(s). This verifiability property was motivated by the unlinkability requirement – in particular it prevents the issuer from fingerprinting users by having a unique key per user. In the following constructions we consider a weaker verifiability property which guarantees that the user receives a valid token issued with the "committed" key(s) or a random value. The new verifiability property implies that the issuer *can* distinguish valid tokens from invalid tokens since the user cannot verify whether she has a valid token. Hence, the issuer is able to distinguish users for which it issued valid tokens from users holding invalid tokens. This means that the issuer can partition the users into sets that receive valid and invalid tokens and he will be able to identify these sets – this affects the success probability of the adversary in the unlinkability game, which we discuss in the corresponding proofs in Theorems 19 and 21.

7.1 Privacy Pass Without ZK

We start with our new construction for the functionality of Privacy Pass. The change that we make is that the user blind its token hash $\mathsf{H}_t(t)$ using both multiplicative and additive mask. The additional additive mask can be removed during the unblinding if the issuer used the correct secret key. Otherwise, the additive mask is sent to a random group element, which



```
Usr(pp = $X, t \in \{0,1\}^\lambda$)                      Sig(pp, sk := $x$)

AT.Usr$_0$(pp, $t$)
  $r, \rho \leftarrow_\$ \mathbb{Z}_p^*$
  $T := \mathsf{H}_t(t)$
  $T' := r \cdot (T - \rho G)$                     $\xrightarrow{T'}$
  return $(T', (\mathsf{pp}, r, \rho, t, T'))$
                                                           AT.Srv$_1$(pp, $x$, $T'$)
                                                             $W' := xT'$
                                 $\xleftarrow{W'}$           return $W'$
AT.Usr$_1$((pp, $r, \rho, t, T'$), $W'$)
  $\sigma := r^{-1}W' + \rho X$
  return $(t, \sigma)$
```

Fig. 11. Token issuance for Construction 4 (Privacy Pass without DLEQ).

makes the whole token indistinguishable from random. Next we provide the detailed construction.

Construction 4 (Privacy Pass without DLEQ) *Let* $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ *be an algorithm that generates a group* $\mathbb{G}$ *of order* p *and outputs a random generator* G*. Let* $(\mathsf{P}_{DLOG1}, \mathsf{V}_{DLOG1})$ *be a proof system for the DLOG relationship defining the language* $\mathcal{L}_{DLOG1}$*.*

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

- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Sample a random invertible value* $x \leftarrow_\$ \mathbb{Z}_p$ *and set* $\mathsf{sk} := x, X := xG$*.*
  - *Generate a proof of knowledge* $\pi \leftarrow \mathsf{P}_{DLOG1}(X, x)$ *and set* $\mathsf{pp} := (X, \pi)$*.*
- $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Fig. 11.*
- *bool* $\leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$*: parse* $\mathsf{sk} = x$*. If* $\sigma = x\mathsf{H}_t(t)$*, return* 1*. Otherwise, return* 0*.*

Correctness. If the issuer is honest, the user computes $\sigma = r^{-1}W' + \rho X = r^{-1}xr(T - \rho G) + \rho X = xT - \rho(xG) + \rho X = xT$.

Security. We prove unforgeability and unlinkability for the construction in Appendix F.

User Verifiability. The protocol presented in the previous section does not enable the user to verify that she has received valid token at the end of an execution. We can enable such verifiability for any number of tokens at the cost of one additional issuance interaction between the user and the issuer. In particular, let $(t_i, \sigma_i)_{i\in[m]}$ be m token that the user has been issued. She sends a token issuance request $T' = \sum_{i\in[m]} c_i \mathsf{H}_t(t_i)$ where $c_i \leftarrow\!\!\$\, \mathbb{Z}_p$ for $i \in [m]$. Let σ be the issuer's response after unblinding, then the user checks that $\sigma = \sum_{i\in[m]} c_i\sigma_i$.

If the issuer was honest, then $\sigma_i = x\mathsf{H}_t(t_i)$ and

$$\sigma = x\left(\sum_{i\in[m]} c_i \mathsf{H}_t(t)\right) = \sum_{i\in[m]} c_i(x\mathsf{H}_t(t_i)) = \sum_{i\in[m]} c_i\sigma_i\,.$$

Next, we argue that if $\sigma = \sum_{i\in[m]} c_i\sigma_i$, then the issuer could be cheating on any of the m token executions only with negligible probability. We will prove this by induction on m. Let $m = 1$, then we have $\sigma = c_1\sigma_1$ and at the same time $\sigma_1 \neq x\mathsf{H}_t(t_1)$. By the unlinkability argument above we know that σ_1 and hence $c_1\sigma_1$ are uniformly distributed. Hence, the adversary has only negligible probability to guess the value σ.

Now, let us assume that the statement holds for $m \leq k$ and we will be prove it for $m = k+1$. We have $\sigma = \sum_{i\in[m]} c_i\sigma_i$ and at the same time there exists an index j such that $\sigma_j \neq x\mathsf{H}_t(t_j)$. If there is an index k such that $\sigma_k = x\mathsf{H}_t(t_k)$, then $\sigma - x\mathsf{H}_t(t_k) = \sum_{i\in[m]\setminus\{k\}} c_i\sigma_i$ and $j \in [m]\setminus\{k\}$, which contradicts the induction assumption. Therefore, it must be the case that $\sigma_i \neq x\mathsf{H}_t(t_i)$ for all $i \in [m]$. However, by the arguments in the unlinkability proof, we know that all σ_i's, and hence $\sum_{i\in[m]} c_i\sigma_i$, will be distributed uniformly at random. Hence, the adversary has only negligible probability in guessing the value of σ, which concludes the inductive proof.

7.2 PMBTokens without ZK

The challenge to generalizing the construction of the previous section to the setting of private metadata is that the user should not find out what metadata bit value the issuer used and hence which public key it should use when unblinding. Our solution will be to have the user run the unblinding with both keys where only one of the resulting values will be a valid token under the corresponding key for the bit value while the other unblinded value will be completely random. When we do this we need to be careful that the issuer who also generates the public keys should not be able to make the two unblinded values correlated, which would open an avenue for fingerprinting.

To guarantee that the unblinded value with the public key that does not correspond to the embedded private metadata bit is random and hence it is independent of the other unblinded value, even in the case when the issuer is misbehaving, we will need to have that the user generate two independent blinded values which it sends in its first message. The issuer will be using only one of the received blinded tokens to sign and embed his metadata bit, however, the user will be unblinding the message coming from the issuer using two independent sets of blinding parameters, which would thwart the issuer from embedding correlations.

Construction 5 (PMBTokens without DLEQOR) *Let $\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ be algorithm that generates a group $\mathbb{G}$ of order p and outputs two distinct random generators G and H (in particular, the discrete log of H with respect to G is unknown). 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 $(\mathsf{P}_{DLOG2}, \mathsf{V}_{DLOG2})$ be a proof system for the DLOG relationship defining the language $\mathcal{L}_{DLOG2}$.*

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

– $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(1^\lambda)$*:*
  - *Run $\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ to obtain group parameters. Γ will be an implicit input to all other algorithms.*
  - *Sample four random invertible values $x_0, y_0, x_1, y_1 \leftarrow_\$ \mathbb{Z}_p^*$, and compute $X_0 = x_0 G + y_0 H$, $X_1 = x_1 G + y_1 H$. Generate two proofs of knowledge $\pi_0 \leftarrow \mathsf{P}_{DLOG2}(X_0, (x_0, y_0))$ and $\pi_1 \leftarrow \mathsf{P}_{DLOG2}(X_1, (x_1, y_1))$.*
  - *Set $\mathsf{sk} := ((x_0, y_0), (x_1, y_1))$, $\mathsf{pp} := (X_0, X_1, \pi_0, \pi_1)$.*

– $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}) \rangle$ *– the anonymous token issuance protocol is defined in Fig. 12.*

– *bool* $\leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$*: return* 1.

– $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, (t, \sigma))$*:*
  - *Parse $\sigma = (S_0, S_1, W_0, W_1)$ and $\mathsf{sk} = ((x_0, y_0), (x_1, y_1))$.*
  - *If $W_0 = x_0 \mathsf{H}_t(t) + y_0 S_0$ and $W_1 \neq x_1 \mathsf{H}_t(t) + y_1 S_1$, return 0*
  - *If $W_0 \neq x_0 \mathsf{H}_t(t) + y_0 S_0$ and $W_1 = x_1 \mathsf{H}_t(t) + y_1 S_1$, return 1*
  - *Else, return $\perp$.*

```
Usr(pp = $(X_0, X_1, \pi_0, \pi_1), t)$                         Sig(pp, sk = $((x_0, y_0), (x_1, y_1)), b)$

AT.Usr$_0$(pp, $t$)
  $T \coloneqq \mathsf{H}_t(t)$
  $\forall d \in \{0,1\}$
    $r_d, \rho_d \leftarrow\$ \mathbb{Z}_p^*$
    $T'_d \coloneqq r_d \cdot (T - \rho_d G)$
  return $((T'_0, T'_1), (\mathsf{pp}, \{r_d, \rho_d, T'_d\}_{d=0,1}, t))$

                 $\xrightarrow{T'_0, T'_1}$

                                               AT.Srv$_1$(pp, $x$, $T'$)
                                                 $s \leftarrow\$ \{0,1\}^\lambda$
                                                 $S'_b \coloneqq \mathsf{H}_s(T'_b, s)$
                                                 $W' \coloneqq x_b T'_b + y_b S'_b$
                                                 return $(s, W')$

                 $\xleftarrow{s, W'}$

AT.Usr$_1$((pp, $\{r_d, \rho_d, T'_d\}_{d=0,1}, t), (s, W'))$
  $\forall d \in \{0,1\}$
    $S_d \coloneqq r_d^{-1}\mathsf{H}_s(T'_d, s) + \rho_d H$
    $W_d \coloneqq r_d^{-1} \cdot W' + \rho_d X_d$
  $\sigma \coloneqq (S_0, S_1, W_0, W_1)$
  return $(t, \sigma)$
```

Fig. 12. Token issuance for Construction 5 (PMBTokens without DLEQOR).

Correctness. If the issuer is honest, the user obtains $S_b = r_b^{-1}\mathsf{H}_s(T'_b, s) + \rho_b H$ and

$$\begin{aligned}
W_b = r_b^{-1}W' + \rho_b X_b &= r_b^{-1}(x_b T'_b + y_b S'_b) + \rho_b X_b \\
&= r_b^{-1}x_b(r_b(T - \rho_b G)) + y_b r_b^{-1}\mathsf{H}_s(T'_b, s) + \rho_b X_b \\
&= x_b T + y_b r_b^{-1}\mathsf{H}_s(T'_b, s) + \rho_b X_b - \rho_b(x_b G) = x_b T + y_b r_b^{-1}\mathsf{H}_s(T'_b, s) + \rho_b y_b H \\
&= x_b T + y_b(r_b^{-1}\mathsf{H}_s(T'_b, s) + \rho_b H) = x_b T + y_b S_b.
\end{aligned}$$

Security. We provide the proofs for the security properties of the construction in Appendix G.

8 Implementation

We implemented our construction in pure Rust (stable, version `1.41.0`), using the Ristretto group[6] on the top of Curve25519 [?], as provided

[6] https://datatracker.ietf.org/doc/draft-hdevalence-cfrg-ristretto/

Table 2. Benchmarks for our constructions.

Constructions	DLEQ/DLEQOR		Client		Server		
	Prove	Verify	Token Gen.	Unblinding	Key Gen.	Signing	Redemption
Construction 1 (Privacy Pass)	212 µs	180 µs	108 µs	329 µs	80 µs	297 µs	95 µs
Construction 3 (PMBTokens)	680 µs	823 µs	108 µs	922 µs	235 µs	822 µs	241 µs
Construction 4 (Privacy Pass w/o DLEQ)	–	–	166 µs	148 µs	130 µs	75 µs	86 µs
Construction 5 (PMBTokens w/o DLEQOR)	–	–	309 µs	566 µs	323 µs	144 µs	249 µs

by `curve25519-dalek`[7]. Hashing into the group is done with a Elligator 2 map [47] with SHA-512. Using `rust-wasm`[8], we were able to compile the Rust implementation into WebAssembly, and generate blinded tokens from JavaScript in Chromium (version `79.0.3945.130`). Our implementation is not copyrighted and is released in the public domain.

We benchmarked our own implementation on a single thread of an `Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz`, running Ubuntu 18.04.3 LTS (kernel version 4.15.0). They are summarized in Table 2. Proving time for a DLEQ relation is 212 µs; verification is 180 µs. The total time for issuing a Privacy Pass token (Construction 1) is 297 µs; unbliding 329 µs. Generating a blinded token is 108 µs; key generation is 80 µs. The total time for issuing a PMBToken in Construction 3 is 822 µs; unbliding 922 µs. Generating a blinded token is 108 µs; key generation is 235 µs. As expected, Constructions 4 and 5 feature very fast issuance time. When compiled to WebAssembly, we measured 0.6 ms for key generation. Our results are between ten and one thousand faster than the previous implementation proposed in [22] due to the different choice[9] of elliptic curve (NIST P-256) as well as the programming language used.

Acknowledgments

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

[7] https://dalek.rs/
[8] https://rustwasm.github.io/
[9] We comment on the choice of curve and its security in Appendix I.

References

1. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In *CT-RSA*, volume 2020 of *LNCS*, pages 143–158. Springer, 2001.
2. Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In *CT-RSA 2006*, 2006.
3. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
4. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptology*, 16(3):185–215, 2003.
5. Mihir Bellare and Adriana Palacio. GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, volume 2442 of *LNCS*, pages 162–177. Springer, 2002.
6. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *Public Key Cryptography*, volume 6571 of *LNCS*, pages 403–422. Springer, 2011.
7. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In *SCN*, volume 7485 of *LNCS*, pages 95–112. Springer, 2012.
8. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, volume 2567 of *LNCS*, pages 31–46. Springer, 2003.
9. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.
10. Daniel R. L. Brown and Robert P. Gallant. The static diffie-hellman problem. *IACR Cryptology ePrint Archive*, 2004:306, 2004.
11. Jan Camenisch. Efficient and generalized group signatures. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 465–479. Springer, 1997.
12. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.
13. David Cash, Eike Kiltz, and Victor Shoup. The twin diffie–hellman problem and applications. *J. Cryptology*, 22(4):470–504, 2009.
14. David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.
15. David Chaum. Zero-knowledge undeniable signatures. In *EUROCRYPT*, volume 473 of *LNCS*, pages 458–464. Springer, 1990.
16. David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.
17. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, volume 740 of *LNCS*, pages 89–105. Springer, 1992.
18. David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, 1991.
19. Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. Cryptology ePrint Archive, Report 2019/1047, 2019.
21. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, volume 839 of *LNCS*, pages 174–187. Springer, 1994.
22. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, 2018.
23. Adam Everspaugh, Rahul Chatterjee, Samuel Scott, Ari Juels, and Thomas Ristenpart. The pythia PRF service. In *USENIX Security Symposium*, pages 547–562. USENIX Association, 2015.
24. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, volume 4117 of *LNCS*, pages 60–77. Springer, 2006.
25. Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In *SCN*, volume 9841 of *LNCS*, pages 391–408. Springer, 2016.
26. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In *CRYPTO (2)*, volume 9216 of *LNCS*, pages 233–253. Springer, 2015.
27. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures in the algebraic group model. Cryptology ePrint Archive, Report 2019/877, 2019.
28. Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 477–495. Springer, 2014.
29. Essam Ghadafi. Formalizing group blind signatures and practical constructions without random oracles. In *ACISP*, volume 7959 of *LNCS*, pages 330–346. Springer, 2013.
30. Ryan Henry. Efficient zero-knowledge proofs and applications. 2014.
31. Javier Herranz and Fabien Laguillaumie. Blind ring signatures secure under the chosen-target-cdh assumption. In *ISC*, volume 4176 of *LNCS*, pages 117–130. Springer, 2006.
32. Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *ASIACRYPT (2)*, volume 8874 of *LNCS*, pages 233–253. Springer, 2014.
33. Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Threshold partially-oblivious prfs with applications to key management. *IACR Cryptology ePrint Archive*, 2018:733, 2018.
34. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In *EUROCRYPT (3)*, volume 10822 of *LNCS*, pages 456–486. Springer, 2018.
35. Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In *SCN*, volume 6280 of *LNCS*, pages 418–435. Springer, 2010.
36. Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and diffie–hellman problems. *J. Mathematical Cryptology*, 2(4):311–326, 2008.
37. Benoît Libert and Jean-Jacques Quisquater. The exact security of an identity based signature and its applications. *IACR Cryptology ePrint Archive*, 2004:102, 2004.
38. Anna Lysyanskaya and Zulfikar Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography*, 1998.

39. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, volume 740 of *LNCS*, pages 31–53. Springer, 1992.
40. Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.
41. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.
42. Zulfikar Ramzan. Group blind digital signatures : theory and applications, 2013. https://dspace.mit.edu/bitstream/handle/1721.1/80561/43557700-MIT.pdf?sequence=2&isAllowed=y.
43. Claus Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *Information and Communications Security*, 2001.
44. Claus Peter Schnorr. Enhancing the security of perfect blind dl-signatures. *Information Sciences*, 176(10):1305 – 1320, 2006.
45. Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In *TCC*, volume 7194 of *LNCS*, pages 133–150. Springer, 2012.
46. Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In *USENIX Security Symposium*, pages 1556–1571. USENIX Association, 2019.
47. Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In *Financial Cryptography*, volume 8437 of *LNCS*, pages 139–156. Springer, 2014.
48. Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *ACM Conference on Computer and Communications Security*, pages 72–81. ACM, 2007.

A Security Assumptions

A.1 DLOG, DDH, CDH

We recall here the classical discrete logarithm, decisional Diffie–Hellman, and computational Diffie–Hellman assumptions that 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 $X \leftarrow\!\!\$\, \mathbb{G}$, any PPT adversary has negligible advantage in returning $x \in \mathbb{Z}_p$ such that $X = xG$. That is,

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

where $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 13.

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

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

where $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Fig. 13.

```
Game $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$
$\Gamma \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\, \mathbb{Z}_p$;   $X \coloneqq xG$
$y \leftarrow \mathsf{A}(\Gamma, X)$
return $(y = x)$
```

```
Game $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$
$\Gamma, \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$P \leftarrow\!\!\$\, \mathbb{G}$
$a \leftarrow\!\!\$\, \mathbb{Z}_p$;   $A \coloneqq aP$
$b \leftarrow\!\!\$\, \mathbb{Z}_p$;   $B \coloneqq bP$
$C_0 \coloneqq abP$;   $C_1 \leftarrow\!\!\$\, \mathbb{G}$
$b' \leftarrow \mathsf{A}(\Gamma, P, A, B, C_\beta)$
return $b'$
```

```
Game $\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$
$\Gamma \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$P \leftarrow\!\!\$\, \mathbb{G}$
$a \leftarrow\!\!\$\, \mathbb{Z}_p$;   $A \coloneqq aP$
$b \leftarrow\!\!\$\, \mathbb{Z}_p$;   $B \coloneqq bP$
$C \leftarrow \mathsf{A}(\Gamma, P, A, B)$
return $(C = abP)$
```

Fig. 13. The games for discrete logarithm, decisional Diffie–Hellman, and computational Diffie–Hellman.

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 $\mathsf{A} \in \mathsf{PPT}$ 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. 13.

A.2 Discussion on Assumptions

The Chosen-target Diffie–Hellman (CTDH) assumption [8], where an adversary is able to observe some honest "handshakes", has proven itself very useful for a range of applications: Password-Authenticated Key Exchanges [34], signatures [37], and private-set intersection [35]. One can also find it called in the literature as *One-More Diffie–Hellman* (1MDH) assumption [5, 4], 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 $\mathit{\Gamma}$, 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 $1\mathrm{MDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$ 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 [36].

B Okamoto–Schnorr DLEQOR Proofs

In the section we provide the constructions for the DLEQ and DLEQOR arguments of knowledge that we use for our token constructions.

Given a group description $\mathit{\Gamma} := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ and $n = \mathsf{poly}(\lambda)$, we provide a zero-knowledge argument of knowledge for the language

$$\mathcal{L}_n := \left\{ \begin{array}{c} (\mathbf{X} := (X_0, X_1, \ldots, X_{n-1}), T, S, W) \in \mathbb{G}^{n+3} \;: \\ \exists (b, x, y) \in [n] \times \mathbb{Z}_p \times \mathbb{Z}_p \;, \quad \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \end{array} \right\}.$$

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

DLEQOR.V((X, S, T, W), (c, u, 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 $c = \mathsf{H}_c(\Gamma, (\mathbf{X}, T, S, W), \mathbf{K}_0, \dots, \mathbf{K}_n)$
```

Fig. 14. Okamoto–Schnorr proof for DLEQOR.

Note that for $n = 1$, the above language is Okamoto–Schnorr [39] for the DLEQ relation, i.e., $\mathcal{L}_{DLEQ2}$, and for $n = 2$ is $\mathcal{L}_{DLEQOR2}$. Also, when $y = 0$ and $n = 1$, the language is the same of Privacy Pass [22], i.e., $\mathcal{L}_{DLEQ1}$. As such, we only introduce the construction for $\mathcal{L}_n$ below, and one can directly obtain the constructions used throughout the paper.

We present our construction in Fig. 14. 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 $(\Gamma, G_1, \dots, G_m) \in \Gamma \times \mathbb{G}^m$, for $m \geq 0$, to a scalar $c \in \mathbb{Z}_p$. The prover and verifier algorithms are the OR-composition of Okamoto–Schnorr [39]. We present the non-interactive version of the protocol, after applying the Fiat-Shamir reduction, for practical convenience.

The proving algorithm (Fig. 14, 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. The verification algorithm (Fig. 14, 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 $\mathsf{H}_c(\mathbf{X}, T, S, W, \mathbf{K}_0, \dots, \mathbf{K}_n)$.

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 [21]: 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 $y_b \coloneqq (v_b - v'_b)/(c_b - c'_b)$.

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

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

Fig. 15. Batched Okamoto–Schnorr proof for DLEQOR.

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 [30]: 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 9. *Assuming the discrete logarithm is hard for* GrGen*, the* DLEQOR *proof system depicted in Fig. 15 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,

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

where $\mathsf{R}(\mathcal{L}_n)$ denotes the relation associated to the language $\mathcal{L}_n$. 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 $(T_i, S_i, W_i)_{i=0}^{m-1}$. 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 $(\mathbf{X}, T_i, S_i, W_i) \in \mathcal{L}_n$, 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:

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

□

C Security Proofs for Construction 1

C.1 Unforgeability

The one-more unforgeability security notion from Privacy Pass [22] 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 C.2. We therefore prove the following corollary:

Corollary 10. *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 13 and Theorem 15. We detail here the differences:

- In the hybrid, the proof system is with respect to the language $\mathcal{L}_{DLEQ1}$ instead of $\mathcal{L}_{DLEQ2}$.
- In the reduction to chosen-target gap Diffie–Hellman, B sets $\mathsf{pp} := A$, and answers the signing queries with whatever HELP answers and the simulated proof. At the end of the game, it outputs whatever A outputs.

This shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQ1},\mathsf{A}}(\lambda).$$

and concludes the proof. □

C.2 Equivalence of CTDH and 1MD

In [22], Privacy Pass is proved 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 $\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ 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 11. *Chosen-Target Diffie–Hellman for* GrGen *holds if and only if One-More Decryption Security for* $\mathsf{Elg}[\mathsf{GrGen}]$ *holds. More precisely, for all* $\ell \in \mathbb{N}$, $\mathrm{CTDH}_{\mathsf{GrGen},A,\ell}(\lambda)$ *is equivalent to* $1\mathrm{MD}_{\mathsf{Elg}[\mathsf{GrGen}],A,\ell}(\lambda)$.

Proof. Let $\ell \in \mathbb{N}$. We start by proving that 1MD $\Longrightarrow$ CTDH. Let $\mathsf{A} \in \mathsf{PPT}$ be an adversary for $1\mathrm{MD}_{\mathsf{Elg}[\mathsf{GrGen}],\mathsf{A},\ell}(\lambda)$. We use it to construct an adversary B for the game $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$. The adversary B receives as input a group description Γ and a group element $X \in \mathbb{G}$. Additionally, it has access to two oracles: a TARGET oracle and a HELP oracle. It start

```
Game $1\mathrm{MD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$
$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$(x, X) \leftarrow \mathsf{Elg.KeyGen}(\Gamma)$
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 $\mathsf{B}(\Gamma, X)$ for $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$
for $i \in [\ell+1]$ :
    $Y_i \leftarrow \textsc{Target}(i)$;   $R_i \leftarrow\!\$\, \mathbb{G}$
    $c_i \coloneqq (C_i, D_i) \coloneqq (Y_i, Y_i + R_i)$
// $\textsc{Dec}(i) \coloneqq Y_i + R_i - \textsc{Help}(Y_i)$
$(M_0, \ldots, M_\ell) \leftarrow \mathsf{A}^{\textsc{Dec}}(X, c_0, \ldots, c_\ell)$
for $i \in [\ell+1]$ :   $Z_i \coloneqq Y_i + R_i - M_i$
return $(i, Z_i)_{i=0}^\ell$
```

Fig. 16. Game for one-more decryption (left) and reduction to chosen-target Diffie–Hellman (right).

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 $(C_i, D_i) \coloneqq (Y_i, Y_i + R_i)$. During its execution, the adversary A may ask for ℓ decryption queries. We answer to those queries returning $M_i \coloneqq (Y_i + R_i) - Z_i$ where $Z_i \coloneqq \textsc{Help}(Y_i)$. At the end of its execution, the adversary B returns $\ell+1$ decryption $(M_0, \ldots, M_\ell) \in \mathbb{G}^{\ell+1}$. The adversary B returns $(i, Z_i \coloneqq Y_i + R_i - M_i)$ for each $i \in [\ell+1]$.

The distribution of each ciphertext is uniform over $\mathbb{G}^2$, exactly as in the game $1\mathrm{MD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$ (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 $1\mathrm{MD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$, it must be the case that it returned M_i for all $i \in [\ell+1]$ such that:

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

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

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

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

The adversary 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 $c_0, \dots, c_\ell$ such that $c_i = (C_i, D_i)$ for all $i \in [\ell+1]$. 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 B returns $Y = \sum_{j=0}^{\ell} a^{\mu(t)j} C_i$;
- for any query of the form $\textsc{Help}(C)$, the adversary 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 $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$, for each $i \in [\ell+1]$, Z_i satisfies:

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

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

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

□

C.3 Unlinkability

In [22], 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 of Theorem 6 by removing all the elements about

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, T')$ in $\mathsf{Hyb}_1$, $\mathsf{Hyb}_2$ .
$s \leftarrow_\$ \{0,1\}^\lambda$
$S' := \mathsf{H}(T', s)$
$W' := xT' + yS'$
$\pi \leftarrow \mathsf{P}_{DLEQ2}((\mathsf{pp}, T', S', W'), (x, y))$   $\pi \leftarrow \mathsf{Sim}_{DLEQ2}((\mathsf{pp}, T', S', W'))$
return $(s, W', \pi)$
```

Fig. 17. Summary of hybrid changes for unforgeability of Construction 2.

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 corollary.

Corollary 12. *Construction 1 provides* 1*-unlinkability according to Definition 3 and assuming a knowledge sound proof system for the language* $\mathcal{L}_{DLEQ1}$.

D Unforgeability of Construction 2

We show that Construction 2 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 additional valid tokens.

D.1 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 $\mathsf{AT.VerValid}$ oracle.

Lemma 13. *Construction 2 is one-more unforgeable, when the adversary does not have access to the* $\mathsf{AT.VerValid}$ *oracle, assuming the hardness of the chosen-target Diffie–Hellman (CTDH) problem.*

Proof. We prove this lemma using a hybrid argument. First, we replace the proving algorithm P_{DLEQ2} by the zero-knowledge simulator Sim_{DLEQ2}, and then show a direct reduction to the chosen-target Diffie–Helman problem. Let $\ell \in \mathbb{N}$.

Hyb_1 This is the game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$. The adversary is provided with the public parameters pp. The adversary has access to the signing oracle

$\mathsf{AT.Srv}_1$ 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 P_{DLEQ2}, we use the zero-knowledge simulator Sim_{DLEQ2} for all signing queries. If there exists an adversary $\mathsf{A} \in \mathsf{PPT}$ whose advantage is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of the DLEQ proof system. Consider the following adversary $\mathsf{B} \in \mathsf{PPT}$ for the game $\mathrm{ZK}^{\beta}_{DLEQ2}(\lambda)$. B generates correctly the public parameters pp, and generate the proofs π's in the signing queries via the $\textsc{Prove}_b$ oracle for the statement $(\mathsf{pp}, T', S', W') \in \mathcal{L}_{DLEQ2}$. At the end of the execution, A returns $\ell + 1$ tokens. If A wins the game, then B outputs 1, otherwise it outputs 0. It follows that, for any adversary $\mathsf{A} \in \mathsf{PPT}$, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQ2},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) \right| .$$

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

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 $\mathsf{pp} \coloneqq A + yH$. Then, it runs $\mathsf{A}(\mathsf{pp})$. Note that pp is distributed as in 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 $\mathsf{AT.Srv}_1$, using the oracles $\textsc{Target}$ and $\textsc{Help}$ from Fig. 1, 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, T')$, B samples $s \leftarrow_{\$} \{0,1\}^{\lambda}$ 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 $\mathsf{AT.Srv}_1$ are identical to the ones of 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 $\{t_i, W_i - yS_i\}_{i=1}^{\ell+1}$.

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 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 $\mathsf{AT.Srv}_1$ at most ℓ times; therefore $\textsc{Help}$ was called at most ℓ times. Finally, this shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) \coloneqq \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQ2},\mathsf{A}}(\lambda).$$

and this concludes the proof. $\square$

D.2 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 likely (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. In Privacy Pass, giving access to the $\mathsf{AT.VerValid}$ oracle enables the adversary to learn if $(X = xG, \mathsf{H}_t(t), W)$ is a DDH tuple for inputs t, W of its choice.

This specific behavior is not unique to our anonymous tokens primitive. 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 in [1, 13];
– when proving the unforgeability of Chaum's undeniable signature scheme in [16, 15, 40];
– when proving the security of blind signatures [9];
– when proving the security of the VOPRF [32], and so on.

Instead, all these schemes are proved under a *gap* problem [40], i.e., a computational problem that gives oracle access to the underlying decision problem.[10] For example, [40] defines the Gap DH problem, which given a

[10] We follow the name usage of [40, 9, 32], but the Gap DH problem is also known under the name of *strong* Diffie–Hellman problem [1, 13].

```
Game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$
$\Gamma \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\, \mathbb{Z}_p$;   $X \coloneqq xG$
$q \coloneqq 0$;   $T = [\,]$
$\{t_i, Z_i\}_{i=1}^{\ell+1} \leftarrow \mathsf{A}^{\textsc{Target},\textsc{Help},\textsc{Ddh}}(\Gamma, X)$
return $(t_i, Y_i) \in T$ all different
          and $xY_i = Z_i \ \ \forall i \in [1, \ell+1]$ and $q \leq \ell$
```

```
Oracle $\textsc{Target}(t_i)$
if $\exists (t_i, Y_i)$ to $T$
  return $Y_i$
else
  $Y_i \leftarrow\!\!\$\, \mathbb{G}^*$
  append $(t_i, Y_i)$ to $T$
  return $Y_i$
```

```
Oracle $\textsc{Help}(Y)$
$q \coloneqq q + 1$
return $xY$
```

```
Oracle $\textsc{Ddh}(t, W)$
return $W \overset{?}{=} x\textsc{Target}(t)$
```

Fig. 18. The Chosen-Target Gap Diffie–Hellman assumption.

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 [8] in *Gap DH* groups [9], that is in groups where CDH is 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 ask for 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 14 (Chosen-target gap Diffie–Hellman.). *The chosen-target gap Diffie–Hellman assumption for the group generator* GrGen *states that all* $\mathsf{A} \in \mathsf{PPT}$*, for all* ℓ*,* A *has negligible advantage in solving CDH on* $\ell + 1$ *target group elements, even if* A *is given access to a DDH oracle, and to a CDH oracle for* ℓ *instances. More formally, for all* ℓ*, for all* $\mathsf{A} \in \mathsf{PPT}$*,*

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

where $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$ *is defined in Fig. 18.*

D.3 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 15. *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 13 is that we need to specify how the queries to $\mathsf{AT.VerValid}(\mathsf{sk}, \cdot)$ are answered to. For any query to $\mathsf{AT.VerValid}(\mathsf{sk}, (t, \sigma))$, the challenger (who knows the secret y) will (1) parse $\sigma = (S, W)$, (2) let $X = W - yS$, and (3) return whatever $\textsc{Ddh}(t, X)$ returns, where $\textsc{Ddh}$ is defined in Fig. 18. This perfectly simulates the oracle. Hence, we obtain that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQ2},\mathsf{A}}(\lambda).$$

and this concludes the proof. □

E Unforgeability and Unlinkability of Construction 3

E.1 Unforgeability

We prove the following theorem using a hybrid argument, similar to the proof of Lemma 13. The key difference is that, during the reduction to CTGDH, B will draw a bit $b \in \{0, 1\}$ which will correspond to its guess on which bit A will succeed for the one-more unforgeability game, and correctly create tokens for the bit $1 - b$.

Theorem 16. *Construction 3 is one-more unforgeable assuming the hardness of the chosen-target (gap) Diffie–Hellman problem.*

Proof. We prove this theorem using a hybrid argument, similar to the proof of Lemma 13. First, we replace the proving algorithm $\mathsf{P}_{DLEQOR2}$ by the zero-knowledge simulator $\mathsf{Sim}_{DLEQOR2}$ (which defines an hybrid Hyb_2), and then show a direct reduction to the chosen-target gap Diffie–Helman problem.

Let ℓ be an integer. We will prove that if there is an adversary $\mathsf{A} \in \mathsf{PPT}$ that has non-negligible advantage $\mathsf{Adv}^{\mathsf{hyb}_2}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ in the hybrid with simulated proofs, then we can construct an adversary $\mathsf{B} \in \mathsf{PPT}$ that has non-negligible advantage in the $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ game.

Assuming the existence of A, we construct B as follows. First, B draws a bit $b \in \{0, 1\}$ which will correspond to its guess on which bit A will succeed for the one-more unforgeability game. B receives the group description and $A \in \mathbb{G}$ as input, samples $y_b, x_b, y_{1-b} \leftarrow \mathbb{Z}_p$ invertible, and computes $X_b \coloneqq A + y_b H$ and $X_{1-b} \coloneqq x_{1-b} G + y_{1-b} H$. With overwhelming probability it holds that $X_0 \neq X_1$. Then, it sets $\mathsf{p} = (X_0, X_1)$ and it runs $\mathsf{A}(\mathsf{pp})$. Note that pp is distributed as in Hyb_2.

We need now to specify how B answers oracle queries. The adversary B overrides the queries to the random oracles H_t, $\mathsf{AT.Srv}_1$, and $\mathsf{AT.ReadBit}$, using the oracles $\textsc{Target}$, $\textsc{Help}$, and $\textsc{Ddh}$ from Fig. 18, 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b', T')$, the adversary B proceeds as follows:
  - it samples $s \leftarrow_\$ \{0,1\}^\lambda$ and defines $S' \coloneqq \mathsf{H}_s(T', s)$;
  - If $b' = 1-b$, it computes $W' = x_{1-b}T' + y_{1-b}S'$; else (i.e., $b' = b$), it invokes the oracle $Z \coloneqq \textsc{Help}(T')$ and sets $W' = Z + y_b S'$.
  - It simulates the proof π, and returns (s, W', π).
- to any query of the form $\mathsf{AT.ReadBit}(\mathsf{sk}, (t, \sigma))$, the challenger (who knows the secrets y_b, x_{1-b}, y_{1-b}) will (1) parse $\sigma = (S, W)$, (2) let $X = W - y_b S$, (3) define $bool_b \coloneqq \textsc{Ddh}(t, X)$ and $bool_{1-b} \coloneqq (W = x_{1-b}\mathsf{H}_t(t) + y_{1-b}S)$, and (4) return $\perp$ if $bool_b = bool_{1-b}$, or then 1 if $bool_1$, or then 0 if $bool_0$.

All other random oracle queries are left unchanged. First, note that the distributions of H_t, $\mathsf{AT.Srv}_1$, and $\mathsf{AT.ReadBit}$ are identical to the ones of 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 $\{t_i, W_i - y_b S_i\}_{i=1}^{\ell+1}$.

Finally, it follows that the adversary B wins the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ every time that A wins and produced a forgery on b. This shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) \coloneqq \mathsf{Adv}^{\mathsf{hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLEQ2},\mathsf{A}}(\lambda) + \mathsf{negl}(\lambda)\,.$$

and this concludes the proof. □

E.2 Unlinkability

We prove that Construction 3 is **2**-unlinkable instead of 1-unlinkable (cf. Definition 3), which means that the probability that an adversary can guess which of m tokens not redeemed yet is upper-bounded by $2/m + \mathsf{negl}(\lambda)$. Indeed, they key idea is that the adversary can now embed different private metadata bits during the issuances, halving its search space at most.

Theorem 17. *Construction 3 provides* 2*-unlinkability (Definition 3) assuming the hardness of DDH and a knowledge sound proof system for the language* $\mathcal{L}_{DLEQOR2}$.

Proof. The theorem trivially holds for $m = 1, 2$. Let $m > 2$. The theorem can be proved by a sequence of hybrids similar to those of the proof of Theorem 6. Since the user algorithms are the same (besides the DLOE-QOR instead of DLEQ proof) as in Construction 2, we explicit the key difference in the reduction below, and refer to Fig. 8 and the proof of Theorem 6 for the rest of the proof.

The key idea is as follows. We will first extract all the witnesses from the proofs $\pi_i, i \in U$, and hence we will be able to partition the set U in two: the indices with a witness starting with the bit 0 and the indices with a witness starting with the bit 1. We will then sample a biased bit b depending on the probability $|U_1|/|U|$, then sample j, k in U_b and perform the same steps as in the proof of Theorem 6, which proves that the adversary can only guess j with probability $1/|U_b|$. Hence, the probability of success $\mathfrak{p}$ of the adversary will be upper bounded by

$$\mathfrak{p} = \sum_{\substack{b'=0,1\\ U_{b'}\neq\emptyset}} \Pr[b' = b] \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda) = \sum_{\substack{b'=0,1\\ U_{b'}\neq\emptyset}} \frac{|U_{b'}|}{|U|} \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda)$$
$$= \sum_{\substack{b'=0,1\\ U_{b'}\neq\emptyset}} \frac{|U_{b'}|}{k_1 - k_2} \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda) \leq \frac{2}{k_1 - k_2} + \mathsf{negl}(\lambda) \leq \frac{2}{m} + \mathsf{negl}(\lambda)\ .$$

□

F Security Proofs for Construction 4

In this section, we prove that Construction 4 is one-more unforgeable and unlinkable.

F.1 Unforgeability

The unforgeability of the scheme follows from the unforgeability of Privacy Pass since here the user receives strictly less information than in the Privacy Pass construction.

Theorem 18. *Construction 4 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

F.2 Unlinkability

We will formally argue that the each token is either a valid token or is a uniformly distributed value. Indeed, note that there exists $P \in \mathbb{Z}_p$ such that $\sigma' = xT' + P$. If $P = 0$ and the issuer was honest, then $\sigma = x\mathsf{H}_t(t)$. Otherwise, $\sigma = x\mathsf{H}_t(t) + r^{-1}P$. The value T' is uniformly distributed since ρ is chosen at random, and then σ is uniformly distributed since r is chosen at random.

Theorem 19. *Construction 4 provides 2-unlinkability according to Definition 3 and assuming a knowledge sound proof system for the language $\mathcal{L}_{DLOG1}$.*

```
Game UNLINK_{AT,A,m}(λ) in Hyb_1, Hyb_2
    $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
    $(\mathsf{st}, \mathsf{pp} = (X, \pi)) \leftarrow \mathsf{A}(\Gamma)$
    $x \leftarrow \mathsf{Ext}_{DLOG1}(\pi)$
    if not $R(x, X)$ then abort
    $k_1 := 0; k_2 := 0; U := \emptyset$
    $(\mathsf{st}, \{W'_i\}_{i \in U}) \leftarrow \mathsf{A}^{\text{USR}_0(), \text{USR}_1(\cdot,\cdot)}(\mathsf{st})$
    if $U = \emptyset$ then return $0$
    $j \leftarrow\$ U$;  $U := U \setminus \{j\}$
    $\mathsf{token}_j := \mathsf{AT.Usr}_1(\mathsf{st}_j, W'_j)$
    $\forall i \in U, \mathsf{token}_i := \mathsf{AT.Usr}_1(\mathsf{st}_i, W'_i)$
    $\phi \leftarrow \mathcal{S}_U$
    $j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \{\mathsf{token}_{\phi(i)}\}_{i \in U})$
    return $k_1 - k_2 \geq m$ and $j' = j$
```

```
Oracle USR_0()
    $k_1 := k_1 + 1$
    $t_{k_1} \leftarrow\$ \{0,1\}^\lambda$
    $(T'_{k_1}, \mathsf{st}_{k_1}) \leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t_{k_1})$
    $U := U \cup \{k_1\}$
    return $(k_1, T'_{k_1})$
```

```
Oracle USR_1(j, W')
    if $j \notin U$ then return $\perp$
    $\mathsf{token} \leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_j, W')$
    $U := U \setminus \{j\}$
    $k_2 := k_2 + 1$
    return token
```

```
AT.Usr_0(pp = (X, π), t)) in Hyb_1, Hyb_4
    $r, \rho \leftarrow\$ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r(T - \rho G)$
    $T' \leftarrow\$ \mathbb{G}$
    $\mathsf{st} := (\mathsf{p}, r, \rho, t)$
    return $(T', \mathsf{st})$
```

```
AT.Usr_1(pp, r, ρ, t), W') in Hyb_1, Hyb_3 , Hyb_5
    $(X, \pi) := \mathsf{pp}$
    $\sigma := r^{-1}W' + \rho X$
    $P := W' - xT'$
    $\sigma := x\mathsf{H}_t(t) + r^{-1}P$
    if $P \neq 0$ then $\sigma \leftarrow\$ \mathbb{G}$
    return $(t, \sigma)$
```

Fig. 19. Summary of hybrid changes for unlinkability for Construction 4.

Proof. We formally argue that the each token is either a valid token or is a uniformly distributed value. We do this with a sequence of hybrids presented in Fig. 19. We argue that the transitions between the hybrids in Fig. 19 are indistinguishable as follows:

Let m be an integer.

Hyb_1 The first hybrid is the unlinkability game.

Hyb_2 In this hybrid, we use the knowledge extractor on the public key. Since the extractor fails only with negligible probability, this hybrid is indistinguishable from the previous one.

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

Hyb_3 In this hybrid we compute the value σ in $\mathsf{AT.Usr}_1$ in a different but equivalent way. First, note that since the challenger knows x, it can define $P \coloneqq W' - xT'$. Then, we have

$$\begin{aligned}\sigma &= r^{-1}W' + \rho X = r^{-1}(xT' + P) + \rho X \\ &= r^{-1}(xr(\mathsf{H}_t(t) - \rho G) + P) + \rho X \\ &= x\mathsf{H}_t(t) + r^{-1}P.\end{aligned}$$

The distribution is identical to the previous hybrid.

Hyb_4 In this hybrid, we sample T' uniformly at random, which results in the same distributions since the ρ is sampled at random and is used only in the computation of this specific T'.

Hyb_5 In this hybrid, if $P \neq 0$, we sample σ at random. Since r is sampled at random and is only used in this specific computation, the resulting distribution is the same.

Now, we have that the tokens can be of two types $(t, x\mathsf{H}_t(t))$ or (t, R), where tokens of the same type are indistinguishable for the adversary. Let U_0, U_1 be the sets of tokens from each type issued during the unlinkability game, such that $U_0 \cup U_1$ is a partition of U.

$$\begin{aligned}\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)\right] &= \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) \mid j \in U_d\right] \Pr[j \in U_d] \\ &\leq \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \frac{1}{|U_d|}\frac{|U_d|}{|U|} + \mathsf{negl}(\lambda) \leq \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \frac{1}{k_1 - k_2} + \mathsf{negl}(\lambda) \\ &\leq \frac{2}{m} + \mathsf{negl}(\lambda)\,.\end{aligned}$$

□

G Security Proofs for Construction 5

In this section, we prove that Construction 5 is one-more unforgeable, unlinkable, and provide privacy for the metadata bit.

G.1 Unforgeability

Theorem 20. *Construction 5 is one-more unforgeable assuming the hardness of the chosen-target gap Diffie–Hellman problem.*

Proof. We prove this lemma using a hybrid argument. First, we replace the proving algorithm by the zero-knowledge simulator, and then show a reduction to the chosen-target gap Diffie–Hellman problem.

Let ℓ be an integer.

Hyb_1 This is the game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$: the adversary is provided with the public parameters $\mathsf{pp} = (X, \pi)$. 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 proofs). At the end of its execution, it outputs $\ell + 1$ tokens for the same bit b'.

Hyb_2 This hybrid replaces the way the zero-knowledge proof is generated: instead of using the proving algorithm, we use the zero-knowledge simulator.

If there exists an adversary $\mathsf{A} \in \mathsf{PPT}$ whose advantage is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of the proof system. Indeed, we call the algorithm $\textsc{Prove}_\beta$ to generate the proofs. If A wins the game, then B outputs 1, otherwise it outputs 0. We have

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLOG2},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) \right|$$

We will now 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 gap Diffie–Hellman game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$.

Assuming the existence of A, we construct B as follows. B first guesses the bit b on which the adversary will succeeds. It receives the group description and $A \in \mathbb{G}$ as input, samples $y_b, x_{1-b}, y_{1-b} \leftarrow \mathbb{Z}_p$ invertible, and computes $X_b := A + y_b H$, $X_{1-b} := x_{1-b} G + y_{1-b} H$, and simulates the proofs π_0, π_1. Then, it runs $\mathsf{A}(\mathsf{pp} = (X_0, X_1, \pi_0, \pi_1))$. Note that pp is distributed as in 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$ in the following way.

- For any query to the oracle $\mathsf{H}_t(t)$, the adversary B invokes the oracle $\textsc{Target}(t)$ and returns whatever it returns;
- For any query $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b', (T'_0, T'_1))$, it proceeds as follows.
  - If $b' = 1 - b$, it correctly follows the issuance protocol described in Fig. 12, i.e., it samples $s \leftarrow\!\!\$\ \{0,1\}^\lambda$, define $S'_{1-b} := \mathsf{H}_s(T'_{1-b}, s)$, compute $W' = x_{1-b} T'_{1-b} + y_{1-b} S'_{1-b}$. It then answers (s, W') to A.

  - If $b' = b$, it samples $s \leftarrow_\$ \{0,1\}^\lambda$ and defines $S'_b := \mathsf{H}_s(T'_b, s)$. Then it invokes the oracle $Z := \textsc{Help}(T'_b)$, defines $W' = Z + y_b S'_b$. Finally, it returns (s, W').

– For any query $\mathsf{AT.ReadBit}(\mathsf{sk}, (t, \sigma = (S_0, S_1, W_0, W_1)))$, B uses the $\textsc{Ddh}$ and its secret scalars $y_b, x_{1-b}, y_{1-b} \in \mathbb{Z}_p$ to check if $\textsc{Ddh}(t, W_b - y_b S_b)$ and if $W_{1-b} = x_{1-b}\mathsf{H}_t(t) + y_{1-b}S_b$, and answers correctly depending on the values of the booleans.

All other random oracle queries are left unchanged. First, note that the distributions of H_t and $\mathsf{AT.ReadBit}$ are identical to the ones of 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 $\{t_i, W_i - y_b S_i\}_{i=0}^{\ell}$.

We claim that the adversary B wins the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ every time that A wins if he guesses the bit correctly. By the winning condition of game Hyb_2, A won if and only if all t_i are different and $W_i = x\textsc{Target}(t_i) + y_{b'}S_i$ where x is the unique element of $\mathbb{Z}_p$ such that $A = xG$ and b' is the bit on which the forgery happens. Furthermore, by the winning condition, A only called the signing oracle $\mathsf{AT.ReadBit}$ at most ℓ times; therefore $\textsc{Help}$ was called at most ℓ times.

Finally, this shows that

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) := \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \frac{1}{2}\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLOG2},\mathsf{A}}(\lambda).$$

and this concludes the proof. □

G.2 Unlinkability

Theorem 21. *Construction 5 provides* 3*-unlinkability according to Definition 3, assuming the hardness of DDH and a knowledge sound proof system for the language $\mathcal{L}_{DLOG2}$.*

Proof. We will formally argue that for an adversary (issuer) who knows the secret key $\mathsf{sk} := ((x_0, y_0), (x_1, y_1))$, the values T'_0, T'_1 received during the issuance execution, the values S_0, S_1 from the signature are indistinguishable from random, and at most one of the value W_0, W_1 is distinguishable from random and is of the form $W_b = x_b\mathsf{H}_t(t) + y_b S_b$.

We present in Fig. 20 the sequence of hybrids that transition from an honest execution on the user side of a token issuance to an execution where all the values in the token are random except the relation that allows to read out one value for the embedded private metadata bit.

Let m be an integer.

```
Game UNLINK$_{\mathsf{AT,A},m}(\lambda)$ in Hyb$_1$, Hyb$_2$
$\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{st}, \mathsf{pp} = (X_0, X_1, \pi_0, \pi_1)) \leftarrow \mathsf{A}(\Gamma)$
$(x_0, y_0) \leftarrow \mathsf{Ext}_{DLOG1}(\pi_0)$
if not $R((x_0, y_0), X_0)$ then abort
$(x_1, y_1) \leftarrow \mathsf{Ext}_{DLOG1}(\pi_1)$
if not $R((x_1, y_1), X_1)$ then abort
$k_1 := 0; k_2 := 0; U := \emptyset$
$(\mathsf{st}, \{W'_i\}_{i \in U}) \leftarrow \mathsf{A}^{\textsc{Usr}_0(), \textsc{Usr}_1(\cdot,\cdot)}(\mathsf{st})$
if $U = \emptyset$ then return 0
$j \leftarrow\!\$\, U$;   $U := U \setminus \{j\}$
$\mathsf{token}_j := \mathsf{AT.Usr}_1(\mathsf{st}_j, W'_j)$
$\forall i \in U, \mathsf{token}_i := \mathsf{AT.Usr}_1(\mathsf{st}_i, W'_i)$
$\phi \leftarrow \mathcal{S}_U$
$j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \{\mathsf{token}_{\phi(i)}\}_{i \in U})$
return $k_1 - k_2 \geq m$ and $j' = j$
```

```
$\mathsf{AT.Usr}_1((\mathsf{pp}, \{r_d, \rho_d, T'_d\}_{d=0,1}, t), (s, W'))$ in Hyb$_1$, Hyb$_3$ , Hyb$_4$ , $\boxed{\text{Hyb}_5}$
$(X_0, X_1, \pi_0, \pi_1) := \mathsf{pp}$
$S_0 = r_0^{-1}\mathsf{H}_s(T'_0, s) + \rho_0 H$;   $S_0 \leftarrow\!\$\, \mathbb{G}$;   $W_0 := r_0^{-1}W' + \rho_0 X_0$
$S_1 = r_1^{-1}\mathsf{H}_s(T'_1, s) + \rho_1 H$   $\boxed{S_1 \leftarrow\!\$\, \mathbb{G};}$   $W_1 := r_1^{-1}W' + \rho_1 X_1$
$P_0 := W' - x_0 T'_0 - y_0 \mathsf{H}_s(T'_0, s)$;   $W_0 := x_0 \mathsf{H}_t(t) + y_0 S_0 + r_0^{-1} P_0$
$P_1 := W' - x_1 T'_1 - y_1 \mathsf{H}_s(T'_1, s)$;   $W_1 := x_1 \mathsf{H}_t(t) + y_1 S_1 + r_1^{-1} P_1$
$\sigma := (S_0, S_1, W_0, W_1)$
return $(t, \sigma)$
```

```
$\mathsf{AT.Usr}_1((\mathsf{pp}, \{r_d, \rho_d, T'_d\}_{d=0,1}, t), (s, W'))$ in Hyb$_8$ , Hyb$_9$
$(X_0, X_1, \pi_0, \pi_1) := \mathsf{pp}$
$S_0 \leftarrow\!\$\, \mathbb{G}$
$S_1 \leftarrow\!\$\, \mathbb{G}$
$P_0 := W' - x_0 T'_0 - y_0 \mathsf{H}_s(T'_0, s)$;   $W_0 := x_0 \mathsf{H}_t(t) + y_0 S_0 + r_0^{-1} P_0$
$P_1 := W' - x_1 T'_1 - y_1 \mathsf{H}_s(T'_1, s)$;   $W_1 := x_1 \mathsf{H}_t(t) + y_1 S_1 + r_1^{-1} P_1$
if $P_0 \neq 0$ then $W_0 \leftarrow\!\$\, \mathbb{G}$
if $P_1 \neq 0$ then $W_1 \leftarrow\!\$\, \mathbb{G}$
$\sigma := (S_0, S_1, W_0, W_1)$
return $(t, \sigma)$
```

```
Oracle $\textsc{Usr}_0()$
$k_1 := k_1 + 1$
$t_{k_1} \leftarrow\!\$\, \{0,1\}^\lambda$
$(T'_{k_1}, \mathsf{st}_{k_1}) \leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t_{k_1})$
$U := U \cup \{k_1\}$
return $(k_1, T'_{k_1})$
```

```
$\mathsf{AT.Usr}_0(\mathsf{pp} = (X, \pi), t))$ in Hyb$_1$, Hyb$_6$ , Hyb$_7$
$T := \mathsf{H}_t(t)$
$r_0, \rho_0 \leftarrow\!\$\, \mathbb{Z}_p^*$
$r_1, \rho_1 \leftarrow\!\$\, \mathbb{Z}_p^*$
$T'_0 := r_0(T - \rho_0 G)$
$T'_1 := r_1(T - \rho_1 G)$
$T'_0 \leftarrow\!\$\, \mathbb{G}$
$T'_1 \leftarrow\!\$\, \mathbb{G}$
$\mathsf{st} := (\mathsf{pp}, \{r_d, \rho_d\}_{d=0,1}, t)$
return $((T'_0, T'_1), \mathsf{st})$
```

```
Oracle $\textsc{Usr}_1(j, W')$
if $j \notin U$ then return $\perp$
$\mathsf{token} \leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_j, W')$
$U := U \setminus \{j\}$
$k_2 := k_2 + 1$
return token
```

Fig. 20. Summary of hybrid changes for unlinkability of Construction 5.

Hyb$_1$ This is the execution where the adversary interacts with an honest user side for the token issuance.

Hyb$_2$ In this hybrid, we run the knowledge extractor on the public keys in the public parameters. This hybrid is indistinguishable from the previous since the extractor succeeds with all but negligible probability.

$$2\mathsf{Adv}^{\mathrm{kdns}}_{\mathsf{A,GrGen}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT,A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT,A}}(\lambda) \right| .$$

Hyb$_3$ In this hybrid, we compute W_0 and W_1 in a different but equivalent way in $\mathsf{AT.Usr}_1$. First, note that since the challenger knows x_0, y_0, x_1 and y_1, it can define $P_d := W' - x_d T'_d - y_d \mathsf{H}_s(T'_d, s)$ for $d = 0, 1$. Then, for $d = 0, 1$, we have

$$\begin{aligned} W_d &= r_d^{-1}W' + \rho_d X_d = r_d^{-1}(x_d T'_d + y_d \mathsf{H}_s(T'_d, s) + P_d) + \rho X_d \\ &= r_d^{-1}(x_d r_d(\mathsf{H}_t(t) - \rho_d G) + y_d \mathsf{H}_s(T'_d, s) + P_d) + \rho_d X_d \\ &= x_d \mathsf{H}_t(t) - \rho_d x_d G + y_d r_d^{-1} \mathsf{H}_s(T'_d, s) + r_d^{-1} P_d + \rho_d x_d G + \rho_d y_d H \\ &= x_d \mathsf{H}_t(t) + y_d \left(r_d^{-1}\mathsf{H}_s(T'_d, s) + \rho_d H\right) + r_d^{-1} P_d \\ &= x_d \mathsf{H}_t(t) + y_d S_d + r_d^{-1} P_d. \end{aligned}$$

Hyb_4 In this hybrid, we sample S_0 uniformly at random. We show that if there is an adversary B that distinguishes Hyb_3 and Hyb_4, then we can construct an adversary that breaks DDH.
Let (P, aP, bP, cP) be a DDH challenge. We set $G = P$ and $H = aP$. We will use the self-reducibility of DDH: for each call to $\mathsf{AT.Usr}_1$, we sample $\gamma, \gamma' \leftarrow_\$ \mathbb{Z}_p^*$ and set $\rho_0 G = \gamma bP + \gamma' P$, which is used in the computation of T'_0 that is the only other element that depends on ρ_0. Then, we set $\rho_0 H = \gamma cP + \gamma' aP$ and use it for the computation of S_0. Now if $cP = abP$, then the execution coincides with Hyb_3, and if cP is a random element, then the execution is Hyb_4. Hence,

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

Hyb_5 In this hybrid, we sample S_1 uniformly at random. As before, we can show that if there is an adversary B that distinguishes Hyb_4 and Hyb_5, then we can construct an adversary that breaks DDH.

$$\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{A},\mathsf{GrGen}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_5}_{\mathsf{AT},\mathsf{A}}(\lambda)\right|.$$

Hyb_6 In this hybrid, we sample T'_0 uniformly at random. Since ρ_0 is sampled at random and this is the only place where it is used Hyb_5 and Hyb_6 have exactly the same distribution.

Hyb_7 In this hybrid, we make an analogous change sampling T'_1 uniformly at random. The distribution in this hybrid is the same as in the previous since ρ_1 is sampled at random and used only in the computation of T'_1.

Hyb_8 If $P_0 \neq 0$, we sample W_0 uniformly at random. Since r_0 is sampled at random and only used for the computation of W_0 the resulting distributions of Hyb_7 and Hyb_8 are the same.

Hyb_9 We analogously replace $r_1^{-1}P_1$ with a random value of $P_1 \neq 0$. As above the change does not change the distribution of values.

We argue that it cannot be the case that both $P_0 = 0$ and $P_1 = 0$. Indeed, if there is an adversary B that generated W' such that $W' = x_0 T_0' + y_0 S_0'$ and $W' = x_1 T_1' + y_1 S_1'$, it follows that $(r_0 x_0 - r_1 x_1)\mathsf{H}_t(t) + y_0 \mathsf{H}_s(T_0', s) - y_1 \mathsf{H}_s(T_1', s) = 0$, and the adversary needs to find a s so that the previous equation is true. This can only happen with negligible probability since H_s is modeled as a random oracle. Considering Hyb_9 the view of the adversary for tokens can one of three types

$$(t, S_0, S_1, x_0\mathsf{H}_t(t)+y_0 S_0, W_1), (t, S_0, S_1, W_0, x_1\mathsf{H}_t(t)+y_1 S_1), \text{ or } (t, S_0, S_1, W_0, W_1)$$

where all variables S_0, S_1, W_0, W_1 are uniformly distributed. Let $U_1, U_2, U_3 \subset U$ be the the indices of tokens that are in each of these three forms. The above hybrids that tokens that have the same type are indistinguishable for the adversary. Therefore,

$$\begin{aligned}
\Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)\right] &= \sum_{\substack{d=1,2,3\\ U_d \neq \emptyset}} \Pr\left[\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda) \mid j \in U_d\right] \Pr[j \in U_d] \\
&\leq \sum_{\substack{d=1,2,3\\ U_d \neq \emptyset}} \frac{1}{|U_d|}\frac{|U_d|}{|U|} + \mathsf{negl}(\lambda) \leq \sum_{\substack{d=1,2,3\\ U_d \neq \emptyset}} \frac{1}{k_1 - k_2} + \mathsf{negl}(\lambda) \\
&\leq \frac{3}{m} + \mathsf{negl}(\lambda)\,.
\end{aligned}$$

□

G.3 Privacy of the Metadata Bit

Theorem 22. *Construction 5 provides privacy for the metadata bit (Definition 4) assuming the hardness of DDH.*

Proof. We consider the sequence of hybrids presented in Fig. 21 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_1 This is the game $\mathrm{PMB}^{0}_{\mathsf{AT},\mathsf{A}}(\lambda)$: here, the adversary is provided the public parameters $\mathsf{pp} := (X_0, X_1, \pi_0, \pi_1)$. The adversary has access to the signing oracle for a bit of its choosing, and a challenge oracle

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b}, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_3(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T'_{\hat{b}}, s)$ was queried then abort
$S'_{\hat{b}} := \mathsf{H}(T'_{\hat{b}}, s)$
$W' := x_{\hat{b}} T'_{\hat{b}} + y_{\hat{b}} S'_{\hat{b}}$
return $(s, W')$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$ in $\mathsf{Hyb}_1(\lambda)$, $\mathsf{Hyb}_3(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T'_b, s)$ was queried then abort
$S'_b := \mathsf{H}(T'_b, s)$
$W' := x_b T'_b + y_b S'_b$
return $(s, W')$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b}, T')$ in $\mathsf{Hyb}_4(\lambda), \mathsf{Hyb}_5(\lambda), \mathsf{Hyb}_6(\lambda)$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T'_{\hat{b}}, s)$ was queried then abort
$S'_{\hat{b}} := \mathsf{H}(T'_{\hat{b}}, s)$
$W' := x_{\hat{b}} T'_{\hat{b}} + y_{\hat{b}} S'_{\hat{b}}$
return $(s, W')$
```

```
Oracle $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$ in $\mathsf{Hyb}_4(\lambda)$, $\mathsf{Hyb}_5(\lambda)$, $\boxed{\mathsf{Hyb}_6(\lambda)}$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T'_b, s)$ was queried then abort
$S'_b := \mathsf{H}(T'_b, s)$
$W' := x_b T'_b + y_b S'_b$
$y' \leftarrow\$ \mathbb{Z}_p^*$;  $W' := x_b T'_b + y' S'_b$
$W' := x_{1-b} T'_{1-b} + y' S'_{1-b}$
$\boxed{W' := x_{1-b} T'_{1-b} + y_{1-b} S'_{1-b}}$
return $(s, W')$
```

Fig. 21. Summary of the proof for privacy of the metadata bit of Construction 5.

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_2 This hybrid replaces the way zero-knowledge proofs are generated: instead of using the proving algorithm P_{DLOG2}, we use the zero-knowledge simulator Sim_{DLOG2}.
If there exists an adversary $\mathsf{A} \in \mathsf{PPT}$ whose output is different between the two games, then it is possible to construct an adversary for the underlying zero-knowledge of the proof system: consider the adversary $\mathsf{B} \in \mathsf{PPT}$ for the game $\mathrm{ZK}^{\beta}_{DLOG2}(\lambda)$ that generates X_0, X_1 as per $\mathsf{AT.KeyGen}(1^\lambda)$ and uses the $\textsc{Prove}_\beta$ oracle for the statements $(X_0) \in \mathcal{L}_{DLOG2}$ and $(X_1) \in \mathcal{L}_{DLOG2}$. At the end of its execution, A (and so B) return a guess b'.
If the $\textsc{Prove}_\beta$ oracle outputs proofs via P_{DLOG2}, the game is identical to Hyb_1, else the game is identical to Hyb_2. It follows that, for any adversary $\mathsf{A} \in \mathsf{PPT}$, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in DLEQOR, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\mathcal{L}_{DLOG2},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{AT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A}}(\lambda).\right|$$

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'_b, 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 $\mathsf{A} \in \mathsf{PPT}$ making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles H_s, $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \cdot, \cdot)$, or $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$, 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 $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$ we construct W' as $x_b T'_b + y' S'_b$ instead of $x_b T' + y_b S'_b$.

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:

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

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, where B programs the RO the response to H_s as we discuss next. We replace queries to the signing oracles:

– for any query $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, b, T')$, 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 sample $\alpha, \beta \leftarrow_\$ \mathbb{Z}_p$ and we program the random oracle on $\mathsf{H}_s(T'_b, s) = S'_b$ to reply with $(\beta_i B + \alpha_i P)$, for some $\alpha_i, \beta_i \leftarrow_\$ \mathbb{Z}_p$. Then, B computes $W' := x_b T' + \beta_i C + \alpha_i A$. B returns (s, W')

– for any query $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b} = b, T')$, after sampling $s \leftarrow_\$ \{0,1\}^\lambda$, we program the random oracle on $\mathsf{H}(T'_b, s) = S'_b$ to reply with $\alpha_i H$ for some $\alpha_i \leftarrow_\$ \mathbb{Z}_p$. B computes $W' := x_b T'_b + \alpha_i A$. It returns (s, W').

– any query to $\mathsf{AT.Srv}_1(\mathsf{pp}, \mathsf{sk}, \hat{b} = 1 - b, T')$ 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. Therefore, every time that A's output is different between the two hybrids, 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 $W' := x_b T'_b + y' S'$, 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 $x_b T'_b$, i.e., uniform) as long as $S'_b \neq 0G$. Therefore, we change once again the way we compute W', swapping b with $1-b$: in this hybrid, $W' := x_{1-b} T'_{1-b} + y' S'_{1-b}$. For the above remarks, the two games can be distinguished only if S'_b pr S'_{1-b} is the identity element, which happens with probability $2/p$.

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 $\mathsf{Hyb}_3 \rightarrow \mathsf{Hyb}_4$. 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 $\mathsf{AT}.\mathsf{Srv}_1(\mathsf{pp}, \mathsf{sk}, b, \cdot)$ 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 $\mathsf{P}_{DLEQOR2}$ (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 $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$

$$\mathsf{Adv}^{\mathrm{hb}}_{\mathsf{AT},\mathsf{A}}(\lambda) \leq \frac{q(q-1)}{2^\lambda} + \frac{2}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 4\mathsf{Adv}^{\mathrm{zk}}_{DLOG2}(\lambda)$$

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 $\lambda = \lfloor \log_2 p \rfloor$. $\square$

H PMBTokens with 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.

```
Usr(pp, $t$)                                                    Sig(pp, sk, $b$)

  AT.Usr$_0$(pp, $t$)
    $r \leftarrow\!\!\$\ \mathbb{Z}_p^*$
    $T \coloneqq \mathsf{H}_t(t)$
    $T' \coloneqq r^{-1} \cdot T$
    return $(T', (\mathsf{pp}, r, t, T'))$

                              $\xrightarrow{T'}$

                                                    AT.Srv$_1$(pp, sk, $b$, $T'$)
                                                      $(X_0, X_1, \tilde{X}) \coloneqq \mathsf{pp}$
                                                      $((x_0, y_0), (x_1, y_1), (\tilde{x}, \tilde{y})) \coloneqq \mathsf{sk}$
                                                      $s \leftarrow\!\!\$\ \{0,1\}^\lambda$
                                                      $S' \coloneqq \mathsf{H}_s(T', s)$
                                                      $W' \coloneqq x_b T' + y_b S'$
                                                      $\pi \leftarrow \mathsf{P}_{DLEQOR2}((X_0, X_1, T', S', W'), (x_b, y_b))$
                                                      $\tilde{W}' \coloneqq \tilde{x} T' + \tilde{y} S'$
                                                      $\tilde{\pi} \leftarrow \mathsf{P}_{DLEQ2}((\tilde{X}, T', S', \tilde{W}'), (\tilde{x}, \tilde{y}))$
                                                      return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

                              $\xleftarrow{s, W', \tilde{W}', \pi, \tilde{\pi}}$

  AT.Usr$_1$((pp, $r, t, T'$), $(s, W', \tilde{W}', \pi, \tilde{\pi})$)
    $(X_0, X_1, \tilde{X}) \coloneqq \mathsf{pp}$
    $S' \coloneqq \mathsf{H}_s(T', s)$
    if not $\mathsf{V}_{DLEQOR2}((X_0, X_1, T', S', W'), \pi)$
      or not $\mathsf{V}_{DLEQ2}((\tilde{X}, T', S', \tilde{W}'), \tilde{\pi})$ then
      return $\perp$
    $S \coloneqq rS'$
    $W \coloneqq rW'$
    $\tilde{W} \coloneqq r\tilde{W}'$
    $\sigma \coloneqq (S, W, \tilde{W})$
    return $(t, \sigma)$
```

Fig. 22. Token issuance for Construction 6 (anonymous token with private metadata bit and validity verification).

Construction 6 *Let $\Gamma \coloneqq (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$ 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 $(\mathsf{P}_{DLEQ2}, \mathsf{V}_{DLEQ2})$ (resp. $(\mathsf{P}_{DLEQOR2}, \mathsf{V}_{DLEQOR2})$) be a proof system for the DLEQ (resp. DLEQOR) relationship defining the language $\mathcal{L}_{DLEQ2}$ (resp. $\mathcal{L}_{DLEQOR2}$).*

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

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

- *Sample four random invertible values* $x_0, x_1, y_0, y_1 \leftarrow_{\$} \mathbb{Z}_p^*$*, and set* $X_0 := x_0 G + y_0 H$ *and* $X_1 := x_1 G + y_1 H$*. Restart if* $X_0 = X_1$*.*
- *Sample two random invertible values* $\tilde{x}, \tilde{y} \leftarrow_{\$} \mathbb{Z}_p^*$*, and set* $\tilde{X} = \tilde{x}G + \tilde{y}H$*.*
- *Set* $\mathsf{sk} := ((x_0, y_0), (x_1, y_1), (\tilde{x}, \tilde{y}))$ *and* $\mathsf{pp} := (X_0, X_1, \tilde{X})$*.*

– $(t, \sigma) \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sig}(\mathsf{pp}, \mathsf{sk}, b) \rangle$ *– the anonymous token singing protocol is defined in Fig. 22.*

– *bool* $\leftarrow \mathsf{AT.VerValid}(\mathsf{sk}, t, \sigma)$*:*
- *Parse* $\sigma = (S, W, \tilde{W})$*.*
- *If* $\tilde{W} = \tilde{x}\mathsf{H}_t(t) + \tilde{y}S$*, return* 1*. Otherwise, return* 0*.*

– $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, t, \sigma)$*:*
- *Parse* $\sigma = (S, W, \tilde{W})$*.*
- *If* $W = x_0\mathsf{H}_t(t) + y_0 S$ *and* $W \neq x_1\mathsf{H}_t(t) + y_1 S$*, return* 0
- *If* $W \neq x_0\mathsf{H}_t(t) + y_0 S$ *and* $W = x_1\mathsf{H}_t(t) + y_1 S$*, return* 1
- *Else, return* $\perp$*.*

H.1 Unforgeability

Theorem 23. *Construction 6 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 6 in the same way as the reduction in the proof of Theorem 16 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 6. □

H.2 Unlinkability

Theorem 24. *Construction 6 provides* 2*-unlinkability according to Definition 3 assuming the hardness of DDH, a knowledge sound proof system for the language* $\mathcal{L}_{DLEQ2}$*, and a knowledge sound proof system for the language* $\mathcal{L}_{DLEQOR2}$*.*

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 6 in the same way as the reduction in the proof of Theorem 17 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 6. □

H.3 Privacy of Metadata Bit

Theorem 25. *Construction 6 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 8. 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. □

I Choice of Curve

Because our constructions rely on the presence of a so-called static DH oracle, special care must be taken when choosing the group, as in fact stronger cryptanalytic attacks are at disposal for the adversary. These are Brown-Gallant [10] and Cheon's attack [19], as already studied independently by Taylor Campbell[11] in the context of Privacy Pass, by Thomas et al. [46] in the context of OPRFs, and by Chiesa et al. [20] in the context of SNARKs. Both attacks exploit the smoothness of $p \pm 1$, where p is the order of the group; if $p \pm 1$ is smooth, such as in the case of NISTP224, the security drops to $O(2^{88.5})$ for $O(2^{47})$ queries [46]. In the case of Ristretto, $p-1$ is not smooth. The $p-1$ attacks have best complexity $O(\sqrt{p/d} + \sqrt{d})$ and demand a number of *sequential* queries proportional to $\max(d, p/d)$, where d is a divisor of $p \pm 1$. This concretely provides a best attack of complexity $O(2^{124})$. The case $p+1$ has instead complexity $O(\sqrt{p/d}+d)$, and requires a similar number of oracle queries. In this case, for Ristretto the security drops to $O(2^{94})$ with $O(2^{64})$ sequential queries. Therefore, due to the large number of sequential queries needed to mount the attack, even for adversaries with close proximity, this attack can be mitigated with appropriate key rotation mechanisms.

[11] https://mailarchive.ietf.org/arch/msg/cfrg/YDVS5Trpr6suig_VCFEOH6SOn8Q