Anonymous Tokens with Private Metadata Bit

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

[1] Google, New York, USA
[2] École Normale Supérieure, CNRS, PSL University, Paris, France,
[3] Inria, Paris, France
[4] Recurse Center, New York, USA

`{benkreuter,tancrede,marianar}@google.com`
`michele.orru@ens.psl.eu`

April 21, 2022

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 a lightweight, single-use anonymous trust token 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 generalizes and extends the functionality of Privacy Pass (PETS'18) with this private metadata bit capability. It is based on the DDH and CTDH assumptions in the random oracle model and provides unforgeability, unlinkability, and privacy for the metadata bit. Both Privacy Pass and PMBTokens rely on non-interactive zero-knowledge proofs (NIZKs). We present new techniques to remove the need for NIZKs, while still achieving 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 [Cha82, CL01]. While we have constructions that support complex statements, 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 of Privacy Pass [DGS+18]. 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 traffic that could drain network resources causing a denial of service (DoS). 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 is the case for users of privacy tools, such as VPNs, Tor, or I2P. Privacy Pass [DGS+18] proposes a solution for this problem, using anonymous tokens as a mechanism to prove trustworthiness of the requests without compromising on user privacy. Since CDNs need to potentially handle millions of requests per second, efficiency of the cryptographic construction is of extreme importance.

In this paper, we consider anonymous tokens that can convey two trust signals, in such 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 be identified as a threat if the issuer stops providing them with tokens. In a real-world system relying on anonymous tokens without private metadata bit, if the issuer stops providing malicious users with tokens, the attacker will know that they have been detected as malicious. In fact, this information could serve as an incentive to corrupt more users, or to train machine learning models that detect which malicious behavior goes un-noticed. Being able to pass on the information whether a user is on an allow or disallow list, and consume it in appropriate ways at redeption time, mitigates such behavior. There has been recent interest in primitives that provide such functionality in standardization bodies such as the IETF and W3C. This includes a recent draft proposal for a *Trust Token API* submitted by Google at the W3C, which calls for a secret metadata bit functionality.[5] Also an IETF working group[6] is discussing standardization of the core protocol of Privacy Pass used by Cloudflare[7] together with extensions including private metadata bit.

In this work, 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 token with a private metadata bit (PMB) embedded in it. The private metadata bit can be read from a token using the secret key held by the issuer, at redemption time. 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. Anonymous token schemes offer the following security properties: unforgeability, unlinkability, and privacy of the metadata bit. *Unforgeability* guarantees that nobody but the issuer can generate new valid tokens. *Unlinkability* 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

Our work includes the following contributions. We formalize the security properties of the primitive of anonymous tokens with private metadata bit. We present a new construction for this primitive, called PMBToken, which extends Privacy Pass (PP) to support private metadata bit, while maintaining competitive efficiency. Further, we introduce new techniques that allow to remove the need for NIZK in the constructions of both Privacy Pass and PMBToken. This simplifies and optimizes the constructions in which the NIZK proof computation is a major bottleneck. The resulting schemes satisfy a weaker unlinkability notion. Finally, we implement all the above candidate constructions in Rust, and we summarize the performance of our schemes.

A failed approach and its insight. The starting point of our study is Privacy Pass, which uses the verifiable oblivious PRF (VOPRF) primitive of Jarecki et al. [JKK14] $F_x(t) = x\mathsf{H}_t(t)$ (additively denoted). In the oblivious PRF evaluation mechanism, the user sends to the issuer $r\mathsf{H}_t(t)$ for a randomly selected value r, receives back $rx\mathsf{H}_t(t)$ from which she recovers the output $x\mathsf{H}_t(t)$.[8] Obliviousness is guaranteed by the blinding factor r, which makes the distribution of $r\mathsf{H}_t(t)$ uniform even when knowing t. The PRF output can be verified by the user providing her

[5] See `https://github.com/WICG/trust-token-api#extension-metadata` and `https://web.dev/trust-tokens/`.
[6] See `https://datatracker.ietf.org/wg/privacypass/about/`.
[7] See `https://blog.cloudflare.com/cloudflare-supports-privacy-pass/`.
[8] In Privacy Pass the resulting value $x\mathsf{H}_t(t)$ is used for the derivation of a HMAC key in order to avoid credential hijacking(cf. Appendix D). To simplify our presentation, we skip this step and simply assume that credentials are redeemed over a secure channel.

with a DLEQ proof, which guarantees that $\log_{\mathsf{H}_t(t)}(F_x(t)) = \log_G X$, where G is the base point and $X \coloneqq xG$ is published by the issuer as a public parameter for the scheme.

There is a natural idea to upgrade the above functionality to support a private metadata bit, which is to have 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. If the issuer "signs" two tokens using two different keys, and use them to indicate the different private metadata bit values, then the VOPRF evaluations on the same input t will be the same if they are issued with the same key and will be different if used with different keys. Thus, if a malicious user demands multiple tokens using the same input value t (the issuer, by blindness, has no way of telling), she will be able to distinguish which ones were issued with the same metadata 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 $F_{(x,y)}(t; S) = x\mathsf{H}_t(t) + yS$, 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 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 (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 proof that $X = xG + yH$ and $W = x\mathsf{H}_t(t) + yS$ are computed using the same secret key (x, y). This is similar to Okamoto–Schnorr blind signatures [Oka92], 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 the NIZK. Both Privacy Pass and PMBTokens employ zero-knowledge arguments of knowledge to achieve unlinkability. This approach guarantees that the user can verify that tokens are 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. In section Section 7 we consider a slightly weaker unlinkability guarantee for the user during 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. The user cannot know in advance whether she has a token that will be valid at redemption; incorrectly issued tokens are indistinguishable from honestly-generated ones. If the issuer misbehaves, incorrectly issued tokens will be indistinguishable from random, malformed tokens.

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 that 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. [JKR18], 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. [JKR18], 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 user 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 $T'_d = r_d(\mathsf{H}_t(t) - \rho_d G)$, for $d = 0, 1$, and 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 unforgeability and 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 [DGS+18]. We provide a new proof for unforgeability of Privacy Pass in the presence of a verification oracle based on a different hardness assumption, the *Chosen Target Gap Diffie-Hellman* assumption, which is a formalization of the Chosen Target Diffie-Hellman in a Gap DH group, which was defined by Boneh et al. [BLS01]. In the context of anonymous tokens with private metadata bit, we distinguish a Verify oracle which just simply checks validity of the token, and a Read oracle that returns the *value* of the private metadata bit (which could be 0, 1, or invalid, and in some applications, e.g. blocklisting, 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 access to the Verify oracle, but we crucially require that the adversary *does not get an oracle access that reads the private metadata bit of a token.*

Table 1. Computation and communication costs of our constructions.

Construction	# Multiplications		Communication
	user	issuer	(# elements)
PP (Constr. 1), [DGS+18]	6	3	2
OSPP (Constr. 2)	9	6	2
PMBT (Constr. 3)	15	12	2
PPB (Constr. 4)	4	1	2
PMBTB (Constr. 5)	12	2	3

Efficiency of our constructions. We consider the most expensive computation operation in the above protocols (scalar multiplication) and the largest communication overhead (the number of group elements transferred). We report in Table 1 the efficiency of our constructions. Additionally, the variant of our constructions that supports a 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 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 845 µs and redemption takes 235 µs, while Privacy Pass issuance runs in 303 µs and redemption takes 95 µs. Without the issuance NIZK, PMBTB (Constr. 5) introduces 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.

To simplify the presentation, we present our security proofs in Appendices E, F.1 and G to I, and Appendix J describes a construction where security holds even with a verification oracle.

1.2 Related work

Starting with the work of Chaum [Cha82], 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 [CP92, Oka92, PS00, Sch01, Sch06, FPS19], 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 [FHS15, Bol03]. Partially blind signatures, for which we also have round-optimal constructions [Fis06, SC12, BPV12], 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. The works of Boldyareva [Bol03] and Bellare et al. [BNPS03] achieve round optimal (one round) constructions in the random oracle model under interactive assumptions. These constructions use the same blinding idea as the VOPRF [JKK14] 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 [Fis06, SC12, BFPV11] to constructions in the standard model [GG14, FHS15, FHKS16], but they rely on bilinear groups. This adds complexity to the group instantiations for schemes and computational cost, which we aim to minimize.

Group signatures [CvH91, Cam97, BMW03] allow all the members of a 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 [LR98], 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 [LR98, Ram13, Gha13] require multiple rounds of interaction for the oblivious signing and communication of many group elements.

Abdalla et al. [ANN06] 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 [Cha82]. Davidson et al. [DGS$^+$18] construct a similar private key functionality for anonymous tokens using a VOPRF [JKK14]; it is called Privacy Pass and is the basis of this work.

Everspaugh et al. [ECS$^+$15] introduce partially oblivious PRF, which analogously to blind signatures, allow 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. [JKR18] show how to obtain a threshold variant of the partially oblivious PRF.

The work of Tsang et al. [TAKS07] 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.

In keyed-verification anonymous credentials [CMZ14], the issuer and verifier are the same party. They use an *algebraic* MAC in place of a signature scheme, where the message space is a n-tuple of elements in $\mathbb{Z}_p$ (or in $\mathbb{G}$). They can be used to provide an anonymous token primitive (at a slightly higher cost) but they're overall meant for multi-use credentials. We are not aware of any extension that allows for the embedding of a private metadata bit.

2 Preliminaries

Notation. When sampling the value x uniformly at random from the set S, we write $x \leftarrow_{\$} S$. When sampling the value x from 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 $\{0, \dots, n-1\}$. 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) PPT algorithms A, B 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 $[\![a, \bot]\!] \leftarrow \langle \mathsf{A}, \mathsf{B} \rangle$.

```
Game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$
$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\, \mathbb{Z}_p;\quad X := xG$
$q := 0;\quad \mathcal{Q} := [\,]$
$(t_i, Z_i)_{i\in[\ell+1]} \leftarrow \mathsf{A}^{\textsc{Target},\textsc{Help},\textsc{Ddh}}(\Gamma, X)$
for $i \in [\ell+1]$ :
    if $t_i \notin \mathcal{Q}$ then return 0
    $Y_i := \mathcal{Q}[t_i]$
return $\big(q \le \ell$ and
        $\forall i \ne j \in [\ell+1]\ \ t_i \ne t_j$ and
        $\forall i \in [\ell+1]\ \ xY_i = Z_i\big)$
```

```
Oracle $\textsc{Target}(t)$
if $t \in \mathcal{Q}$ then
    $Y := \mathcal{Q}[t]$
else :
    $Y \leftarrow\!\!\$\, \mathbb{G}$
    $\mathcal{Q}[t] := Y$
return $Y$
```

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

```
Oracle $\textsc{Ddh}(Y, Z)$
return $(Z = x \cdot Y)$
```

Fig. 1. The Chosen-target gap Diffie–Hellman security game.

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

In Appendix A, we define the classical discrete logarithm (DLOG), decisional Diffie–Hellman (DDH), and computational Diffie–Hellman (CDH) assumptions. Here, we define the chosen-target gap Diffie–Hellman (CTGDH) assumption.

Chosen-target gap Diffie–Hellman. The so-called chosen-target Diffie–Hellman (CTDH) assumption [Bol03, HL06] states that any PPT adversary 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. We formalize here its *gap* [OP01] flavor, in which the adversary has, in addition, access to a DDH oracle for arbitrary group elements. Note that the CTDH assumption was originally introduced by Boldyreva [Bol03] in gap DH groups [BLS01], that is, in groups where CDH is hard but DDH is assumed to be easy. In other words, the original definition of CTDH was *already* in groups where the adversary has access to a DDH oracle. Here, we introduce the *chosen-target gap Diffie–Hellman* assumption (CTGDH, that is, the gap version of CTDH) as a security experiment where the adversary is provided a challenge $X \in \mathbb{G}$, and has access to three oracles: the $\textsc{Target}$ oracle, that given as input a string $t \in \{0,1\}^*$, outputs a random group element; the $\textsc{Help}$ oracle, that outputs the CDH of X with an arbitrary group element $Y \in \mathbb{G}$, and the $\textsc{Ddh}$ oracle, that given as input two group elements $(Y, Z) \in \mathbb{G}^2$ returns 1 if and only if (X, Y, Z) is a Diffie–Hellman tuple. We describe the $\textsc{Target}$ oracle in this cumbersome way to ease readability of the security proofs later.

Formally, we say that CTGDH holds for the group generator GrGen if for any PPT adversary A, and any $\ell \ge 0$:

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

```
Game $\mathrm{KSND}_{\mathsf{\Pi,R,A,Ext}}(\lambda)$
$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{\Pi.Setup}(\Gamma)$
$r \leftarrow_{\$} \{0,1\}^{\mathsf{A.rl}(\lambda)}$;  $(\phi, \pi) := \mathsf{A}(\mathsf{crs}; r)$
$w \leftarrow \mathsf{Ext}(\mathsf{td}, r)$
return $(\mathsf{\Pi.Verify}(\mathsf{crs}, \phi, \pi)$ and $\mathsf{R}(\phi, w) = \mathbf{false})$
```

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

```
Oracle $\mathrm{PROVE}(\phi, w)$
if $\mathsf{R}(\phi, w) = \mathbf{false}$ then
    return $\perp$
$\pi_0 \leftarrow \mathsf{\Pi.Prove}(\mathsf{crs}, \phi, w)$
$\pi_1 \leftarrow \mathsf{\Pi.Sim}(\mathsf{crs}, \tau, \phi)$
return $\pi_\beta$
```

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

where $\mathrm{CTGDH}_{\mathsf{GrGen,A},\ell}(\lambda)$ is defined in Figure 1. Note that for $\ell = 0$, the game $\mathrm{CTGDH}_{\mathsf{GrGen,A},0}(\lambda)$ is equivalent to gap CDH. We direct the reader towards Appendix A for more information on this assumption, and how it relates to previous works.

2.2 Non-interactive arguments of knowledge

A non-interactive proof system $\mathsf{\Pi}$ for relation R consists of the following three algorithms:

– $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{\Pi.Setup}(\Gamma)$, the setup algorithm that outputs a common reference string (CRS) crs together with some trapdoor information td.
– $\pi \leftarrow \mathsf{\Pi.Prove}(\mathsf{crs}, \phi, w)$, a prover which takes as input some $(\phi, w) \in \mathsf{R}$ and a CRS crs, and outputs a proof π.
– *bool* $\leftarrow \mathsf{\Pi.Verify}(\mathsf{crs}, \phi, \pi)$ a verifier that, given as input a statement ϕ together with a proof π outputs **true** or **false**, indicating acceptance of the proof.

The proof system $\mathsf{\Pi}$ is a non-interactive zero-knowledge (NIZK) argument of knowledge if it satisfies the following properties:

Completeness. $\mathsf{\Pi}$ is complete if every correctly generated proof verifies. More formally, a proof system $\mathsf{\Pi}$ is *complete* if for any $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$, $\mathsf{crs} \in [\mathsf{\Pi.Setup}(\Gamma)]$ and $(\phi, w) \in \mathsf{R}$:

$$\Pr[\mathsf{\Pi.Verify}(\mathsf{crs}, \phi, \mathsf{\Pi.Prove}(\mathsf{crs}, \phi, w))] = 1 - \mathsf{negl}(\lambda).$$

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

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

where $\mathrm{KSND}_{\mathsf{\Pi,R,A,Ext}}(\lambda)$ is defined in Figure 2 and $\mathsf{A.rl}(\lambda)$ is the randomness length of the adversary A. An *argument of knowledge* is a knowledge-sound proof system. In our proofs, for ease of notation, we will omit sometimes to specify explicitly that the extractor takes as input the coins of the adversary.

Zero Knowledge. A proof system $\mathsf{\Pi}$ for R is *zero-knowledge* if no information about the witness is leaked by the proof, besides membership in the relation. This is formalized by specifying an additional PPT algorithm $\mathsf{\Pi.Sim}$, that takes as input the trapdoor information td and a statement ϕ, and outputs a valid proof π indistinguishable from those generated via $\mathsf{\Pi.Prove}$. Formally, A proof system $\mathsf{\Pi}$ for relation R is zero-knowledge if for any PPT adversary A:

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

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

Throughout this paper, we will assume the existence of the following proof systems, that we summarize here in Camenisch-Stadler notation [Cam97]:

$$\Pi_{\mathsf{DLOG}} \coloneqq \text{NIZK}\{(x) : X = xG\} \tag{1}$$
$$\Pi_{\mathsf{DLEQ}} \coloneqq \text{NIZK}\{(x) : X = xG \ \wedge \ W = xT\} \tag{2}$$
$$\Pi_{\mathsf{DLEQ2}} \coloneqq \text{NIZK}\{(x, y) : X = xG + yH \ \wedge \ W = xT + yS\} \tag{3}$$
$$\Pi_{\mathsf{DLOGAND2}} \coloneqq \text{NIZK}\{(\mathbf{x}, \mathbf{y}) : \forall i \in \{0, 1\} \ X_i = x_i G + y_i H\} \tag{4}$$
$$\Pi_{\mathsf{DLEQOR2}} \coloneqq \text{NIZK}\{(x, y) : \exists i \in \{0, 1\} \ X_i = xG + yH \wedge W = xT + yS\} \tag{5}$$

Eq. (1) and Eq. (4) are discrete logarithm proofs for one, respectively two generators. Eq. (2) and Eq. (3) prove discrete logarithm equality under one, respectively two generators. Finally, Eq. (5) proves discrete logarithm equality for one out of two group elements (in the witness, the index is denoted as $i \in \{0, 1\}$). In Appendix B, we provide a more formal description of the above relations, and efficient instantiations with techniques for batching proofs at issuance time.

3 Anonymous tokens

We describe two flavors of anonymous tokens. The first flavor enables a *user* to obtain a *token* from an *issuer*; the user can later use this token as a trust signal, for one-time authentication. In the second flavor, the issuer has an additional input during the token issuance, a *private metadata bit*, that is hidden within the token. The private metadata bit can later be recovered by the issuer at redemption time. The following definition captures both functionalities; in shaded text, we refer only to the anonymous token with private metadata bit.

Anonymous token. An *anonymous token scheme with private metadata bit* AT consists of the following algorithms:

- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AT.Setup}(1^\lambda)$, the setup algorithm that takes as input the security parameter λ in unary form, and returns a CRS crs and a trapdoor td.
  All the remaining algorithms take crs as their first input, but for notational clarity, we usually omit it from their lists of arguments.
- $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(\mathsf{crs})$, the key generation algorithm that generates a private key sk along with a set of public parameters pp;
- $\sigma \leftarrow \langle \mathsf{AT.User}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b)\rangle$, the token issuance protocol, that involves interactive algorithms $\mathsf{AT.User}$ (run by the user) with input a value $t \in \{0, 1\}^\lambda$, and $\mathsf{AT.Sign}$ (run by the issuer) with input the private key sk and a bit b. At the end of the interaction, the issuer outputs nothing, while the user outputs σ, or $\bot$.
- $bool \leftarrow \mathsf{AT.Verify}(\mathsf{sk}, t, \sigma)$, the verification algorithm that takes as input the private key sk and a token (t, σ). It returns a boolean indicating if the token was valid or not.
- $\mathsf{ind} \leftarrow \mathsf{AT.ReadBit}(\mathsf{sk}, t, \sigma)$, the metadata extraction algorithm that takes as input the private key sk, and a token (t, σ). It returns an indicator $\mathsf{ind} \in \{\bot, 0, 1\}$, which is either the private metadata bit, or $\bot$.

Throughout the rest of this paper, we assume that AT has a one-round signing protocol initiated by the user. Thus, for simplicity, we split the signing algorithms ($\mathsf{AT.Sign}$ and $\mathsf{AT.User}$) into non-interactive algorithms that take as input a message, and the partial state (if any). They will return

```
Game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AT.Setup}(1^\lambda)$
$(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(\mathsf{crs})$
for $b = 0, 1$ : $q_b \coloneqq 0$
$(t_i, \sigma_i)_{i \in [\ell+1]} \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Verify},\textsc{Read}}(\mathsf{crs}, \mathsf{pp})$
return $(\forall b = 0, 1\ q_b \leq \ell$ and
        $\forall i \neq j \in [\ell+1]\ \ t_i \neq t_j$ and
        $\forall i \in [\ell+1]\ \ \mathsf{AT.Verify}(\mathsf{sk}, t_i, \sigma_i) = $ true and
        $\exists b \in \{0,1\} :\ \forall i \in [\ell+1]\ \ \mathsf{AT.ReadBit}(\mathsf{sk}, t_i, \sigma_i) = b)$

Oracle $\textsc{Sign}(b, \mathsf{msg})$
$q_b \coloneqq q_b + 1$
return $\mathsf{AT.Sign}_0(\mathsf{sk}, b, \mathsf{msg})$

Oracle $\textsc{Verify}(t, \sigma)$
return $\mathsf{AT.Verify}(\mathsf{sk}, t, \sigma)$

Oracle $\textsc{Read}(t, \sigma)$
return $\mathsf{AT.ReadBit}(\mathsf{sk}, t, \sigma)$
```

Fig. 3. One-more unforgeability game for the anonymous token scheme AT.

the next message together with the updated state st_i. Concretely, the signing protocol will be composed of the following (non-interactive) algorithms:

– $(\mathsf{usr\_msg}_0, \mathsf{st}_0) \leftarrow \mathsf{AT.User}_0(\mathsf{pp}, t)$;
– $\mathsf{srv\_msg}_1 \leftarrow \mathsf{AT.Sign}_0(\mathsf{sk}, \mathrm{b}, \mathsf{usr\_msg}_0)$;
– $\sigma \leftarrow \mathsf{AT.User}_1(\mathsf{st}_0, \mathsf{srv\_msg}_1)$

We demand that anonymous token schemes satisfies correctness, unforgeability, unlinkability, and privacy of the metadata bit.

Correctness. An anonymous token scheme AT is *correct* if any honestly-generated token verifies and the correct private metadata bit is retrieved successfully. That is, for any $\mathsf{crs} \in [\mathsf{AT.Setup}(1^\lambda)]$, any $(\mathsf{pp}, \mathsf{sk}) \in [\mathsf{AT.KeyGen}(\mathsf{crs})]$, any $t \in \{0,1\}^\lambda$, and $b \in \{0,1\}$:

$$\Pr[\mathsf{AT.Verify}(\mathsf{sk}, t, \langle \mathsf{AT.User}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, \mathrm{b})\rangle) = 1] = 1 - \mathsf{negl}(\lambda), \tag{6}$$
$$\Pr[\mathsf{AT.ReadBit}(\mathsf{sk}, t, \langle \mathsf{AT.User}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b)\rangle) = b] = 1 - \mathsf{negl}(\lambda) \tag{7}$$

Unforgeability. The first security property that we require from an anonymous token is unforgeability, which guarantees that no adversary can redeem more tokens than it is allowed. This is formalized with a standard one-more security game where the adversary can interact with the issuer at most ℓ times, and at the end must output $\ell + 1$ valid tokens. The adversary has also access to a verification oracle for tokens of its choice. In the private metadata bit variant, the adversary can interact with the issuer ℓ times for each private metadata bit, but should not be able to generate $\ell + 1$ valid tokens with the same private metadata.

Definition 1 (One-more unforgeability). *An anonymous token scheme* AT *is* one-more unforgeable *if for any* PPT *adversary* A*, and any* $\ell \geq 0$*:*

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

where $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ *is defined in Figure 3.*

Sometimes, a stronger security notion of unforgeability is desirable, where the adversary can win also by outputting $(t_i, \sigma_i)_{i \in [\ell+1]}$ where some t_i's are the same but the σ_i's are different. The schemes we present were do not provide this type of security.

```
Game $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AT.Setup}(1^\lambda)$
$(\mathsf{st}, \mathsf{pp}) \leftarrow \mathsf{A}(\mathsf{crs})$
$q_0 := 0;\ q_1 := 0;\ \mathcal{Q} := \emptyset$
$(\mathsf{st}, (\mathsf{msg}_i)_{i \in \mathcal{Q}}) \leftarrow \mathsf{A}^{\textsc{User}_0, \textsc{User}_1}(\mathsf{st})$
if $\mathcal{Q} = \emptyset$ then return $0$
// compute a challenge token
$j \leftarrow\$\ \mathcal{Q};\quad \mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$\sigma_j \leftarrow \mathsf{AT.User}_1(\mathsf{st}_j, \mathsf{msg}_j)$
// compute and permute other tokens
for $i \in \mathcal{Q} : \sigma_i \leftarrow \mathsf{AT.User}_1(\mathsf{st}_i, \mathsf{msg}_i)$
$\phi \leftarrow\$\ \mathcal{S}_{\mathcal{Q}}$
$j' \leftarrow \mathsf{A}(\mathsf{st}, (t_j, \sigma_j), (t_{\phi(i)}, \sigma_{\phi(i)})_{i \in \mathcal{Q}})$
return $q_0 - q_1 \geq m$ and $j' = j$

Oracle $\textsc{User}_0()$
$q_0 := q_0 + 1$   // session id
$t_{q_0} \leftarrow\$\ \{0,1\}^\lambda$
$(\mathsf{msg}_{q_0}, \mathsf{st}_{q_0}) \leftarrow \mathsf{AT.User}_0(\mathsf{pp}, t_{q_0})$
$\mathcal{Q} := \mathcal{Q} \cup \{q_0\}$   // open sessions
return $(q_0, \mathsf{msg}_{q_0})$

Oracle $\textsc{User}_1(j, \mathsf{msg})$
if $j \notin \mathcal{Q}$ then
    return $\perp$
$\sigma \leftarrow \mathsf{AT.User}_1(\mathsf{st}_j, \mathsf{msg})$
if $\sigma \neq \perp$ then
    $\mathcal{Q} := \mathcal{Q} \setminus \{j\}$
    $q_1 := q_1 + 1$
return $\sigma$
```

Fig. 4. Unlinkability game for the anonymous token scheme AT. For a set X, $\mathcal{S}_X$ denotes the symmetric group of X.

Unlinkability. This security property is concerned with user anonymity, and guarantees that a malicious issuer cannot link the redemption of a token with a particular execution of the token issuance protocol. More precisely, in κ-unlinkability, if m tokens were issued but not yet redeemed, the adversary cannot link the relative issuance session of a token with probability better than κ/m, even after seeing the remaining $m-1$ tokens in a random order.

Definition 2 (Unlinkability). *An anonymous token scheme* AT *is* κ-unlinkable *if for any* PPT *adversary* A*, and any* $m > 0$*:*

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

where $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},m}(\lambda)$ *is defined in Figure 4.*

Private metadata bit. The last security property protects the private metadata bit in the issued tokens.[9] It guarantees that the private metadata embedded in a token is entirely hidden from anyone who does not possess the private key including the user. More precisely, we require that, even if the adversary corrupts a large number of users, it cannot guess with probability non-negligibly bigger than 1/2 if newly issued tokens have private metadata 0 or 1.

Formally, this is modeled as an indistinguishability game where the adversary has access to two signing oracles: one where it can provide both the message to be signed and the private metadata bit to be used, and one where the adversary chooses only the message (the metadata bit is fixed), and a verification oracle for the validity of the tokens. The adversary's goal is to guess the challenge private metadata bit used.

[9] Consider, e.g., the following practical scenario: the issuer is suspecting that it is targeted by a DoS attack, and decides to tag users that it believes are controlled by a bot using the private metadata bit. The private metadata bit property ensures it is difficult for anyone, but the issuer, to learn how malicious traffic is classified.

```
Game $\mathrm{PMB}^{\beta}_{\mathsf{AT},\mathsf{A}}(\lambda)$
    $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AT.Setup}(1^\lambda)$
    $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(\mathsf{crs})$
    $\beta' \leftarrow \mathsf{A}^{\textsc{Sign},\textsc{Sign}',\textsc{Verify}}(\mathsf{crs}, \mathsf{pp})$
    return $\beta'$

Oracle $\textsc{Sign}(\mathsf{msg})$
    return $\mathsf{AT.Sign}_0(\mathsf{sk}, \beta, \mathsf{msg})$

Oracle $\textsc{Sign}'(b, \mathsf{msg})$
    return $\mathsf{AT.Sign}_0(\mathsf{sk}, b, \mathsf{msg})$

Oracle $\textsc{Verify}(t, \sigma)$
    return $\mathsf{AT.Verify}(\mathsf{sk}, t, \sigma)$
```

Fig. 5. Private metadata bit game for the anonymous token scheme AT.

Definition 3 (Private metadata bit). *An anonymous token scheme* AT *provides* private metadata bit *if for any* PPT *adversary* A*:*

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

where $\mathrm{PMB}^{\beta}_{\mathsf{AT},A}(\lambda)$ *is defined in Figure 5.*

Token hijacking. In our formalization, we do not consider man-in-the-middle adversaries that can steal tokens from honest users. This attack vector, called *token hijacking*, can be mitigated with the use of message authentication codes (MACs). Roughly speaking, instead of sending the entire token (t, σ) over the wire, the user can derive a symmetric key $\mathsf{k} := \mathsf{H}(\sigma)$ to MAC a shared message (e.g., the resource or the URL she's trying to access). The resulting message authentication code is sent together with t to the issuer (and any supplementary randomness that the user needs to recompute σ). We discuss in detail such concerns in Appendix D.

4 Review: Privacy Pass

We start by recalling, using the notation from Section 3, the anonymous token scheme proposed in [DGS+18] (under the name Privacy Pass) and built on top of the verifiable oblivious PRF (VOPRF) "2HashDH-NIZK" [JKK14]. Privacy Pass uses a Schnorr proof in the issuance phase, that we generalize here to any NIZK. Differently from the initial proof of Goldberg et al. [DGS+18], our proof takes into account the presence of a verification oracle, and the knowledge error of the proof system.

Construction 1 (Privacy Pass). Let Π_{DLEQ} be a proof system for relation $\mathsf{R}_{\mathsf{DLEQ}}$; let H_t be a random oracle $\{0,1\}^* \to \mathbb{G}$. The anonymous token scheme PP [DGS+18] is composed of the following algorithms:

– $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PP.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm of the underlying proof system $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQ}}.\mathsf{Setup}(\Gamma)$. Return $\mathsf{crs} := (\Gamma, \mathsf{pcrs})$ and $\mathsf{td} := \mathsf{ptd}$.
– $(X, x) \leftarrow \mathsf{PP.KeyGen}(\mathsf{crs})$: sample a uniformly random element $x \leftarrow_{\$} \mathbb{Z}_p^*$, that will constitute the secret key. Let $X := xG$ be the public parameter. Return (X, x).
– $W \leftarrow \langle \mathsf{PP.User}(X, t), \mathsf{PP.Sign}(x) \rangle$: illustrated in Figure 6.
– $\mathit{bool} \leftarrow \mathsf{PP.Verify}(x, t, W)$: return **true** if $W = x\mathsf{H}_t(t)$; else, return **false**.

Note that this anonymous token protocol is deterministic, i.e., there will exist a unique value $W \in \mathbb{G}$ corresponding to a string $t \in \{0,1\}^\lambda$ that verifies. This property will make difficult to directly extend the construction to support private metadata bit.

```
PP.User($X, t$)                                          PP.Sign($x$)

  PP.User$_0$($X, t$)
    $r \leftarrow_\$ \mathbb{Z}_p^*$
    $T \coloneqq \mathsf{H}_t(t)$
    $T' \coloneqq r^{-1}T$                   $\xrightarrow{T'}$
    return $(T', (X, r, t))$

                                                           PP.Sign$_0$($x, T'$)
                                                             $W' \coloneqq xT'$
                                          $\xleftarrow{W', \pi}$        $\pi \leftarrow \Pi_{\mathsf{DLEQ}}$.Prove($(X, T', W'), x$)
                                                             return $(W', \pi)$

  PP.User$_1$($(X, r, t), (W', \pi)$)
    if not $\Pi_{\mathsf{DLEQ}}$.Verify($(X, T', W'), \pi$) then
        return $\perp$
    return $W \coloneqq rW'$
```

Fig. 6. Token issuance for PP (Construction 1).

Correctness. By correctness of the underlying proof system, at the end of the protocol the user returns $\perp$ only with negligible probability. If the user returns $W \in \mathbb{G}$, then W always satisfies the verification equation, since:

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

Security. PP satisfies both unforgeability and 1-unlinkability.

Theorem 4. *If CTGDH holds for* GrGen *and* Π_{DLEQ} *is a zero-knowledge proof system for relation* R_{DLEQ}*, then* PP[GrGen, Π_{DLEQ}] *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PP},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ}},\mathsf{R}_{DLEQ}}(\lambda).$$

The full proof can be found in Appendix E.1, and follows directly from chosen-target gap Diffie–Hellman for GrGen, and zero-knowledge of Π_{DLEQ}. Consider an adversary A in the game OMUF$_{\mathsf{PP},\mathsf{A}}(\lambda)$. To win the game, A must return $\ell+1$ tokens $(t_i, W_i)_{i\in[\ell+1]}$ such that, for all $i \in [\ell+1]$:

$$\text{(a) } x\mathsf{H}_t(t_i) = W_i, \qquad \text{(b) } \forall j \neq i: \quad t_j \neq t_i \tag{8}$$

During its execution, the adversary A can query at most ℓ times the signing oracle, which given as input $T^* \in \mathbb{G}$ computes the Diffie–Hellman $W = xT^*$, and sends it together with a proof π that W was correctly computed; additionally, A can query the oracle VERIFY(t^*, W^*) that returns 1 if $W^* = x\mathsf{H}_t(t^*)$, that is, if $(X, \mathsf{H}_t(t^*), W^*)$ is a DH tuple. Since Π_{DLEQ} is zero-knowledge, it is possible to simulate the proof π. We are thus left with the game CTGDH (Fig. 1), where VERIFY can be replaced by the oracle DDH, H_t by TARGET, and SIGN by the oracle HELP (together with Π_{DLEQ}.Sim).

```
OSPP.User$(X, t)$                                        OSPP.Sign$((x, y))$

OSPP.User$_0(X, t)$
    $r \leftarrow_\$ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r^{-1} T$
    return $(T', (X, r, t, T'))$
                                  $\xrightarrow{T'}$
                                                         OSPP.Sign$_0((x, y), T')$
                                                             $s \leftarrow_\$ \{0,1\}^\lambda$
                                                             $S' := \mathsf{H}_s(T', s)$
                                                             $W' := xT' + yS'$
                                                             $\pi \leftarrow \Pi_{\mathsf{DLEQ2}}.\mathsf{Prove}((X, T', S', W'), (x, y))$
                                  $\xleftarrow{s, W', \pi}$
                                                             return $(s, W', \pi)$
OSPP.User$_1((X, r, t, T'), (s, W', \pi))$
    $S' := \mathsf{H}_s(T', s)$
    if not $\Pi_{\mathsf{DLEQ2}}.\mathsf{Verify}((X, T', S', W'), \pi)$ then
        return $\perp$
    $S := rS'$
    $W := rW'$
    return $\sigma := (S, W)$
```

Fig. 7. Token issuance for OSPP (Construction 2).

Theorem 5. *Let* GrGen *be a group generator algorithm. If* Π_{DLEQ} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathit{DLEQ}}$*, then* $\mathsf{PP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ}}]$ *is 1-unlinkable.*

We remark that the i-th message sent by PP.User$_0$ is $T'_i = r^{-1}T_i$, for a uniformly random $r_i \in \mathbb{Z}_p^*$. Therefore, T'_i contains no information about T_i or t_i. Additionally, by knowledge soundness of Π_{DLEQ}, it is possible to extract the witness $x \in \mathbb{Z}_p$ used to compute the signatures. With it, the user can compute W_i herself, without ever using the responses from the issuer. It follows that the view of the adversary is limited to random group elements and PP is 1-unlinkable, as long as the proof system is knowledge-sound.

5 Okamoto–Schnorr Privacy Pass

In this section, we describe a novel anonymous token scheme that generalizes PP (Section 4) and allows for *randomized* tokens, which will be an important property when we extend the construction to support private metadata bit (Section 6 and Appendix J). Roughly speaking, while in PP we issue tokens (and DLEQ proofs) using one generator G of $\mathbb{G}$, in this construction we will issue tokens under two generators (G, H), in a similar way to Okamoto–Schnorr [Oka92] signatures. Similarly to Okamoto–Schnorr, it is important here that the discrete logarithm of H base G is unknown. Fixing $y = 0$ in the protocol below, we obtain PP (cf. Section 4).

Construction 2 (Okamoto–Schnorr Privacy Pass). Let GrGen be a group generator algorithm; let Π_{DLEQ2} be a proof system for relation $\mathsf{R}_{\mathsf{DLEQ2}}$; let $\mathsf{H}_t, \mathsf{H}_s$ be two random oracles $\{0,1\}^* \to \mathbb{G}$. We construct an anonymous token scheme OSPP defined by the following algorithms:

– $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{OSPP.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm of the underlying proof system $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQ2}}.\mathsf{Setup}(\Gamma)$. Return $\mathsf{crs} \coloneqq (\Gamma, \mathsf{pcrs})$ and $\mathsf{td} \coloneqq \mathsf{ptd}$.
– $(X, (x,y)) \leftarrow \mathsf{OSPP.KeyGen}(\mathsf{crs})$: sample the secret key $(x,y) \leftarrow\!\!\$\, (\mathbb{Z}_p^*)^2$. Let $X \coloneqq xG + yH$ be the public parameter. Return $(X, (x,y))$.
– $\sigma \leftarrow \langle \mathsf{OSPP.User}(X, t), \mathsf{OSPP.Sign}((x,y)) \rangle$: illustrated in Figure 7.
– $bool \leftarrow \mathsf{OSPP.Verify}((x,y), t, \sigma)$: read $(S, W) \coloneqq \sigma$. Return **true** if $W = x\mathsf{H}_t(t) + yS$; else, return **false**.

Correctness. By correctness of the underlying proof system, the protocol aborts only with negligible probability. If the user returns $\sigma \coloneqq (S, W) \in \mathbb{G}^2$, then σ always satisfies the verification equation, since

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

Security. OSPP satisfies both unforgeability and 1-unlinkability.

Theorem 6. *If CTGDH holds for* GrGen*, and* Π_{DLEQ2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathit{DLEQ2}}$*, then* $\mathsf{OSPP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{OSPP},\ell}(\lambda) \le \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathit{DLEQ2}}}(\lambda).$$

The proof of unforgeability is essentially the same argument of the previous section, with a slightly more careful analysis to deal with the additional element $s \in \{0,1\}^\lambda$. The reduction for CTGDH receives a challenge $A \in \mathbb{G}$, that is now embedded in the public parameters as $X = A + yH$ for some $y \leftarrow\!\!\$\, \mathbb{Z}_p$; the signing oracle computes $W' = \textsc{Help}(T') + yS'$, and the read oracle $\textsc{Ddh}(W - yS)$. The tokens returned by the adversary can be converted in CDH solutions computing $W_i - yS_i$, for all $i \in [\ell + 1]$. The full proof can be found in Appendix F.1.

Theorem 7. *If DDH holds for* GrGen *and* Π_{DLEQ2} *is an argument of knowledge for relation* $\mathsf{R}_{\mathit{DLEQ2}}$*, then* $\mathsf{OSPP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}]$ *is* 1*-unlinkable.*

Similarly to the previous proof, we first notice that in the i-th message, T_i' contains no information about t; additionally, by knowledge soundness of the proof system, W_i must be computed with the same *witness* (x,y) satisfying $X = xG + yH$ (if there exists $(x', y') \neq (x,y)$, then it is possible to construct an adversary for discrete log for GrGen). The core difference now is that we use the same blinding factor r both on S' and W'. The proof hence proceeds in two steps: first, $W \coloneqq rW'$ is computed as $W \coloneqq xT + yS$ with the extracted witness, and next $S \coloneqq rS'$ is replaced by $S \leftarrow\!\!\$\, \mathbb{G}$. This last step can be reduced to DDH: if the adversary manages to distinguish $(T', T = rT', S', S = rS')$ from $(T', T = rT', S', U)$ for $U \leftarrow\!\!\$\, \mathbb{G}$, then it is possible to construct an adversary B for game $\mathrm{DDH}^\beta_{\mathsf{GrGen},\mathsf{B}}(\lambda)$.

```
$\mathsf{PMBT.User}(\mathbf{X}, t)$                                        $\mathsf{PMBT.Sign}((\mathbf{x}, \mathbf{y}), b)$

$\mathsf{PMBT.User}_0(\mathbf{X}, t)$
    $r \leftarrow\!\$\ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r^{-1} T$
    return $(T', (\mathbf{X}, r, t, T'))$

                          $\xrightarrow{T'}$

                                                   $\mathsf{PMBT.Sign}_0((\mathbf{x}, \mathbf{y}), b, T')$
                                                       $s \leftarrow\!\$\ \{0,1\}^\lambda$
                                                       $S' := \mathsf{H}_s(T', s)$
                                                       $W' := x_b T' + y_b S'$
                                                       $\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}((\mathbf{X}, T', S', W'), (x_b, y_b))$
                                                       return $(s, W', \pi)$

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

$\mathsf{PMBT.User}_1((\mathbf{X}, r, t, T'), (s, W', \pi))$
    $S' := \mathsf{H}_s(T', s)$
    if not $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Verify}((\mathbf{X}, T', S', W'), \pi)$ then
        return $\perp$
    $S := rS'$
    $W := rW'$
    return $\sigma := (S, W)$
```

Fig. 8. Token issuance for PMBT (Construction 3).

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. The high-level idea is that we use two different secret keys, one for each private metadata bit. In order to hide which bit is associated with the token, we will use OR proofs (i.e., Π_{DLEQOR2} of Eq. (5)).

Construction 3 (PMBTokens). Let GrGen be a group generator algorithm; let Π_{DLEQOR2} be a proof system for relation $\mathsf{R}_{\mathsf{DLEQOR2}}$; let $\mathsf{H}_t, \mathsf{H}_s$ be two random oracles $\{0,1\}^* \to \mathbb{G}$. We construct an anonymous token scheme PMBT defined by the following algorithms:

– $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PMBT.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm of the underlying proof system $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Setup}(\Gamma)$. Return $\mathsf{crs} := (\Gamma, \mathsf{pcrs})$ and $\mathsf{td} := \mathsf{ptd}$.

– $(\mathbf{X}, (\mathbf{x}, \mathbf{y})) \leftarrow \mathsf{PMBT.KeyGen}(\mathsf{crs})$: let $(\mathbf{x}, \mathbf{y}) \leftarrow\!\$\ (\mathbb{Z}_p^*)^2 \times (\mathbb{Z}_p^*)^2$ be the secret key. Define the public parameters as:

$$\mathbf{X} := \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} := \begin{bmatrix} x_0 G + y_0 H \\ x_1 G + y_1 H \end{bmatrix};$$

restart if $X_0 = X_1$. Return $(\mathbf{X}, (\mathbf{x}, \mathbf{y}))$.

– $\sigma \leftarrow \langle \mathsf{PMBT.User}(\mathsf{pp}, t), \mathsf{PMBT.Sign}(\mathsf{sk}, b)\rangle$: illustrated in Figure 8.
– $\mathit{bool} \leftarrow \mathsf{PMBT.Verify}((\mathbf{x}, \mathbf{y}), t, \sigma)$: return **true**.
– $\mathsf{ind} \leftarrow \mathsf{PMBT.ReadBit}((\mathbf{x}, \mathbf{y}), t, \sigma)$: read $(S, W) \coloneqq \sigma$. Then:
  (a) 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;
  (b) 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;
  (c) else, return $\perp$.

Our construction does *not* provide a (meaningful) implementation of Verify, but only a ReadBit functionality. We elaborate this point in Section 6.1; this construction can be combined with OSPP to provide an actual verification procedure (described in Appendix J).

Correctness. Validity of honestly-generated tokens (cf. Equation (6)) holds perfectly because PMBT.Verify always returns **true**; we focus here on proving that the bit embedded is read correctly with overwhelming probability (Equation (7)). By correctness of the underlying proof system, the protocol aborts only with negligible probability. If the user returns $(S, W) \in \mathbb{G}^2$, then there exists $b \in \{0, 1\}$ such that:

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

The probability that the above equation holds for both $b = 0$ and $b = 1$ (in which case, PMBT.ReadBit returns $\perp$), is statistically negligible. In fact, if:

$$W = x_0\mathsf{H}_t(t) + y_0 S = x_1\mathsf{H}_t(t) + y_1 S,$$

then we have two possibilities:
(a) $y_0 = y_1$, which in turn implies that $x_0\mathsf{H}_t(t) = x_1\mathsf{H}_t(t)$. Because $x_0 \neq x_1$ (by construction of PMBT.KeyGen), this happens only if $\mathsf{H}_t(t)$ is the identity element. This event happens with probability $1/p$;
(b) $y_0 \neq y_1$, which in turn 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 in $\mathbb{G}$ and independently from the terms on the right-hand side. This event happens with probability $1/p$.

Security. In Appendix G, we prove the one-more unforgeability and **2**-unlinkability of PMBT.

Theorem 8. *If CTGDH holds for* GrGen *and* Π_{DLEQ2} *is a zero-knowledge proof system for relation* R_{DLEQ2}*, then* $\mathsf{PMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\text{omuf}}_{\mathsf{PMBT},\ell}(\lambda) \leq 2\mathsf{Adv}^{\text{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\text{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{DLEQ}}(\lambda).$$

The proof is available in Appendix G.1. Consider an adversary A in the game $\text{OMUF}_{\mathsf{PMBT},\mathsf{A}}(\lambda)$. A wins the game if it returns $\ell + 1$ tokens $(t_i, (S_i, W_i))_{i \in [\ell+1]}$ such that there exists a $b \in \{0, 1\}$ satisfying:

$$\text{(a) } \forall i \in [\ell + 1]: \; x_b\mathsf{H}_t(t_i) + y_b S_i = W_i, \qquad \text{(b) } \forall j \neq i: \; t_j \neq t_i. \tag{9}$$

During its execution, A can query $\ell+1$ times the signing oracle for $b=0$, *and* $\ell+1$ times for $b=1$. We claim (in a similar way to Eq. (8)), that A can be used to construct an adversary B that solves CTGDH. We embed the CTGDH challenge $A \in \mathbb{G}$ in one of the two keys: we sample $b^* \leftarrow_\$ \{0,1\}$, and set $X_{b^*} := A + y_{b^*}H$, where $y_{b^*} \leftarrow_\$ \mathbb{Z}_p$. We construct X_{1-b^*} following the key generation algorithm. Queries by A to the oracle $\textsc{Sign}$ are responded in the following way: if the adversary demands issuance for hidden metadata b^*, we use the $\textsc{Help}$ oracle to return $W' = \textsc{Help}(t, T') + y_{b^*}S'$; otherwise B just follows the signing protocol and computes W' using (x_{1-b^*}, y_{1-b^*}). The zero-knowledge proof is simulated. Queries to the oracle $\textsc{Read}$ can still be answered by B with the help of the oracle $\textsc{Ddh}$ available in the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$. Queries to $\textsc{Verify}$ are trivially dealt with, by answering **true**. If at the end of its execution A presents forgeries for the bit $b = b^*$, then by winning condition (a) (cf. Eq. (9)), for all $i \in [\ell+1]$, $(W_i - y_{b^*}S_i)$ is the CDH of the challenge A with $\mathsf{H}_t(t_i)$, which we replace with the CTGDH oracle $\textsc{Target}$ thus presenting $\ell+1$ CDH solutions for the challenge $A \in \mathbb{G}$. If the guess b^* was not correct, then B outputs $\perp$, and we consider the game lost. It follows that B wins the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ half the time that A wins $\mathrm{OMUF}_{\mathsf{PMBT},\mathsf{A},\ell}(\lambda)$.

Theorem 9. *If DDH holds for* GrGen *and* Π_{DLEQOR2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLEQOR2}}$*, then* $\mathsf{PMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQOR2}}]$ *is 2-unlinkable.*

The key idea is that adversary can now embed different private metadata bits at issuance time, at most halving the anonymity set. We use the knowledge extractor to partition the sessions in two buckets: U_0, those associated to the bit 0, and U_1, those with bit 1. We sample a biased $b \in \{0,1\}$ (depending on the distribution of the private metadata bit in the tokens) and select two sessions coming from the same bucket U_b. The probability of success of the adversary will be upper bounded by $2/m + \mathsf{negl}(\lambda)$.

Theorem 10. *If DDH holds for the group generator* GrGen*, and* Π_{DLEQOR2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLEQOR2}}$ *then* $\mathsf{PMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQOR2}}]$ *provides private metadata bit with advantage:*

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBT}}(\lambda) \leq \frac{O(q^2)}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 2\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQOR2}},\mathsf{R}_{\mathsf{DLEQOR2}}}(\lambda),$$

where q is the number of queries the adversary makes to H_s *or* $\textsc{Sign}$*.*

The proof is done by means of a hybrid argument, where the first hybrid is $\mathrm{PMB}^0_{\mathsf{A},\mathsf{PMBT}}(\lambda)$ and the last hybrid is $\mathrm{PMB}^1_{\mathsf{A},\mathsf{PMBT}}(\lambda)$. In the first hybrid, instead of generating the proof using the prover $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$ in the $\textsc{Sign}'$ oracle, we use the zero-knowledge simulator $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}$. The advantage in distinguishing is trivially $\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQOR2}},\mathsf{R}_{\mathsf{DLEQOR2}},\mathsf{B}}(\lambda)$. Then, instead of computing the signature as $W' = x_0T' + y_0S'$, it computes $W' := x_0T' + y'S'$, for some $y' \leftarrow_\$ \mathbb{Z}_p$ sampled after the key generation phase. This hybrid can be shown indistinguishable from the previous one under DDH assumption: $(X - x_0G, S', W' - x_0T') \in \mathbb{G}$ is in fact a DDH triple in the previous hybrid, and a random triple now. Using random self-reducibility of DDH we can answer all queries to $\textsc{Sign}'$ using a single DDH challenge. At this point, we remark that W' is distributed uniformly at random (because $y'S'$ is so) and we can therefore swap x_0 with x_1. A final sequence of hybrid replaces y' with y_1 (again indistinguishable from DDH) and then the simulator with the honest prover.

6.1 Enabling token verification

The anonymous token scheme PMBT (Construction 3) does not provide a meaningful verification algorithm, as it always output **true**. We note that, given two tokens $(t_0, (S_0, W_0))$ and $(t_1, (S_1, W_1))$, if $t_0 = t_1$, then $\big(t^* = t_0 = t_1, (S^* = 2S_0 - S_1, W^* = 2W_0 - W_1)\big)$ is a triple of random elements satisfying $W^* = x_b\mathsf{H}(t^*) + y_b S^*$ only if the same metadata bit b was used. Henceforth, having a verification oracle that checks for the above relation allows an adversary in the game $\mathrm{PMB}^{\beta}_{\mathsf{PMBT},\mathsf{A}}(\lambda)$ to check if two tokens corresponding to the same t were issued with the same private metadata bit.

Instead, we propose to enable such a functionality by combining the PMBT scheme with OSPP, 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 details the design combining Constrs. 2 and 3 in Appendix J.

Jumping ahead, we can also instantiate this design by combining Constructions 4 and 5 that do not use NIZK during issuance. 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.

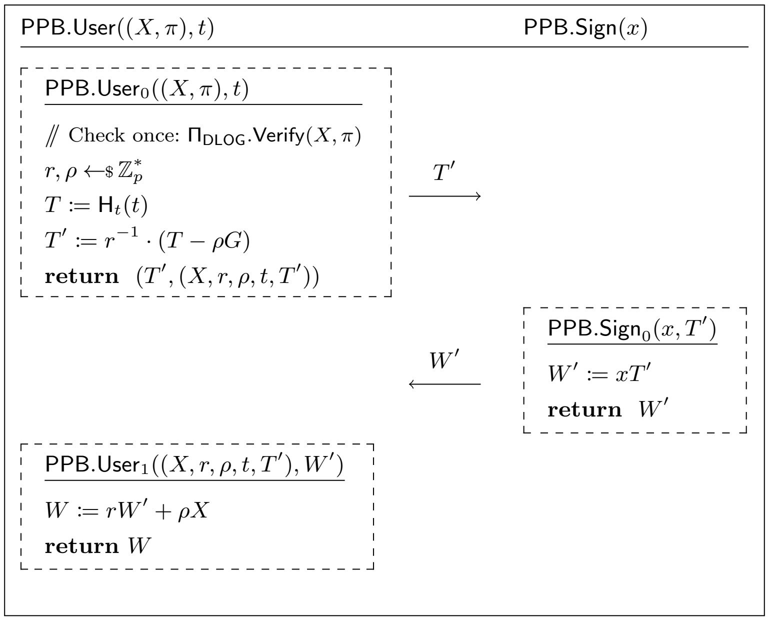
7 Removing the NIZKs during issuance

In this section, we present a general technique for removing the NIZK sent at issuance time in the previous schemes, and replace it with a proof of possession sent only once. We present a formal analysis of it for PP and PMBT (Constructions 1 and 3). We recall that the role of the NIZK is to provide unlinkability for the user, as they can check that the tokens received are consistent with the issuer's public parameters. In particular they prevent the issuer from fingerprinting users by using a unique key per user. We consider a weaker notion of unlinkability, which guarantees that the user either receives a valid token, or a completely random value (unpredictable by the issuer). This implies that the issuer *can* distinguish valid tokens from invalid tokens since the user cannot verify herself whether they have a valid token or not. In other words, the issuer can partition the users into two sets: one that receives valid tokens, and one that receives invalid tokens. The issuer will be able to identify which of these sets a user belongs to, at redemption time. For a more careful analysis on how this affect the success probability of the adversary in the unlinkability game, we refer the reader to Theorems 14 and 22..

In practice, because invalid tokens from a malicious issuers will be uniformly distributed, if *all* clients periodically attempt to redeem a random token, the anonymity set won't change. In fact, since the tokens resulting from the interaction with a malicious issuer are perfectly indistinguishable from tokens transmitted by other users, pragmatically we can achieve the same level of anonymity of the previous protocols.

7.1 PP without issuance NIZK

We start with our new construction for the functionality of Privacy Pass. The change that we make is that the user blinds her token hash $\mathsf{H}_t(t)$ using both multiplicative and additive blinding. The additive part can be removed during the unblinding if the issuer used the correct secret key. Otherwise, the generated token $(t, W) \in \{0,1\}^\lambda \times \mathbb{G}$ will be invalid and distributed uniformly at random.



```
PPB.User((X, π), t)                                  PPB.Sign(x)

  PPB.User_0((X, π), t)
    // Check once: $\Pi_{\mathsf{DLOG}}.\mathsf{Verify}(X, \pi)$
    $r, \rho \leftarrow\$ \mathbb{Z}_p^*$
    $T := \mathsf{H}_t(t)$
    $T' := r^{-1} \cdot (T - \rho G)$
    return $(T', (X, r, \rho, t, T'))$
                                  $\xrightarrow{T'}$
                                                     PPB.Sign_0(x, T')
                                                       $W' := xT'$
                                  $\xleftarrow{W'}$    return $W'$
  PPB.User_1((X, r, ρ, t, T'), W')
    $W := rW' + \rho X$
    return $W$
```

Fig. 9. Token issuance for PPB (Construction 4).

Construction 4 (Privacy Pass without issuance NIZK). Let GrGen be a group generator algorithm; let Π_{DLOG} be a proof system for relation $\mathsf{R}_{\mathsf{DLOG}}$; let H_t be a random oracle $\{0,1\}^* \to \mathbb{G}$. We construct an anonymous token scheme PPB defined by the following algorithms:

- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PPB.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm of the underlying proof system $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLOG}}.\mathsf{Setup}(\Gamma)$. Return $\mathsf{crs} := (\Gamma, \mathsf{pcrs})$ and $\mathsf{td} := \mathsf{ptd}$.
- $((X, \pi), x) \leftarrow \mathsf{PPB.KeyGen}(\mathsf{crs})$: sample the secret key $x \leftarrow\$ \mathbb{Z}_p^*$. Define $\pi \leftarrow \Pi_{\mathsf{DLOG}}.\mathsf{Prove}(\mathsf{pcrs}, X, x)$. Return the public parameters (X, π), and the secret key x.
- $W \leftarrow \langle \mathsf{PPB.User}((X, \pi), t), \mathsf{PPB.Sign}(x) \rangle$: illustrated in Figure 9.
- $\mathit{bool} \leftarrow \mathsf{PPB.Verify}(x, t, W)$: return **true** if $W = x\mathsf{H}_t(t)$; else, return **false**.

Correctness. By correctness of Π_{DLOG}, at the end of the protocol the user returns $\perp$ only with negligible probability. If the user returns $W \in \mathbb{G}$, then W always satisfies the verification equation, since:

$$\begin{aligned} W &= rW' + \rho X \\ &= rxr^{-1}(T - \rho G) + \rho X \\ &= xT - \rho(xG) + \rho X \\ &= xT. \end{aligned}$$

Security. PPB satisfies unforgeability and 2-unlinkability.

Theorem 11. *If CTGDH holds for* GrGen, *and* Π_{DLEQ} *is a zero-knowledge proof system for relation* Π_{DLOG}, *then* PPB *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PPB},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOG}},\mathsf{R}_{\mathsf{DLOG}}}(\lambda).$$

The proof can be found in Appendix H.1. Intuitively, unforgeability must hold because the issuer is sending strictly less information than in PP. The adversary A in game $\mathrm{OMUF}_{\mathsf{PPB},\mathsf{A}}(\lambda)$ takes as input the pair (X, π), where $X \in \mathbb{G}$ is the CTGDH challenge, and π is a DLOG proof. By zero-knowledge of Π_{DLOG}, the proof can be simulated; the adversary is now asked to return $\ell + 1$ pairs (t_i, W_i), all different, such that W_i is the CDH of $(X, \mathsf{H}_t(t_i))$. During its execution, the adversary has at disposal the random oracle H_t (which behaves exactly as the TARGET oracle), the VERIFY oracle, which given as input $(Y, W) \in \mathbb{G}^2$ returns 1 if (X, Y, W) is a DH tuple (just as the DDH oracle in the game for CTGDH), and the SIGN oracle, which computes at most ℓ times the CDH with an arbitrary group element (just as the HELP oracle in the game for CTGDH).

Theorem 12. *Let* GrGen *be a group generator. If* Π_{DLOG} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathsf{DLOG}}$*, then* PPB *is 2-unlinkable with advantage:*

$$\mathsf{Adv}^{\mathrm{unlink}}_{\mathsf{PPB},m}(\lambda) \leq \frac{2}{m} + \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLOG}},\mathsf{R}_{\mathsf{DLOG}}}(\lambda).$$

First of all, we note that by knowledge soundness of Π_{DLOG}, for any adversary A that produces th public parameters (X, π) with $X \in \mathbb{G}$, it is possible to extract a witness $x \in \mathbb{Z}_p$ such that $xG = X$, except with negligible probability $\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLOG}},\mathsf{R}_{\mathsf{DLOG}},\mathsf{A}}(\lambda)$. We use this witness to partition the sessions in two sets, U_0 where W is correctly computed, and U_1 where W isn't. In the latter case, we remark that because the additive blinding $\rho \in \mathbb{Z}_p$ is sampled uniformly at random, in U_1 all tokens are distributed uniformly at random. In a similar way to Theorem 7, we select two sessions k and j from the same U_b (for a $b \in \{0, 1\}$ that sampled at random, following the distribution of the open sessions) and swap them. If k and j are in U_0, unlinkability follows the same reasoning used for proving unlinkability of PP. If k and j are in U_1, then both tokens are uniformly random elements in $\mathbb{G}$ independent from the elements used at issuance time.

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 $(t_i, W_i)_{i \in [m]}$ be m tokens 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 W be the issuer's response after unblinding, then the user checks that $W = \sum_{i \in [m]} c_i W_i$.

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

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

Next, we argue that if $W = \sum_{i \in [m]} c_i W_i$, 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 $W = c_1 W_1$ and at the same time $W_1 \neq x\mathsf{H}_t(t_1)$. By the unlinkability argument above we know that W_1 and hence $c_1 W_1$ are uniformly distributed. Hence, the adversary has only negligible probability to guess the value W.

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

7.2 PMBT without issuance NIZK

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 issuance NIZK). Let GrGen be a group generator algorithm; let Π_{DLOGAND2} be a proof system for the relation $\mathsf{R}_{\mathsf{DLOGAND2}}$; let $\mathsf{H}_t, \mathsf{H}_s$ be two random oracles $\{0,1\}^* \to \mathbb{G}$. We construct an anonymous token scheme PMBTB defined by the following algorithms:

- $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{PMBTB.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLOGAND2}}.\mathsf{Setup}(\Gamma)$. Return $\mathsf{crs} := (\Gamma, \mathsf{pcrs})$ and $\mathsf{td} := \mathsf{ptd}$.
- $((\mathbf{X}, \pi), (\mathbf{x}, \mathbf{y})) \leftarrow \mathsf{PMBTB.KeyGen}(1^\lambda)$: let $(\mathbf{x}, \mathbf{y}) \leftarrow_\$ (\mathbb{Z}_p^*)^2 \times (\mathbb{Z}_p^*)^2$ be the secret key. Define:

$$\mathbf{X} := \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} := \begin{bmatrix} x_0 G + y_0 H \\ x_1 G + y_1 H \end{bmatrix},$$

  and let $\pi \leftarrow \Pi_{\mathsf{DLOGAND2}}.\mathsf{Prove}(\mathsf{pcrs}, \mathbf{X}, (\mathbf{x}, \mathbf{y}))$. The public parameters are $(\mathbf{X}, \pi)$. Return $((\mathbf{X}, \pi), (\mathbf{x}, \mathbf{y}))$.
- $\sigma \leftarrow \langle \mathsf{PMBTB.User}((\mathbf{X}, \pi), t), \mathsf{PMBTB.Sign}((\mathbf{x}, \mathbf{y}))\rangle$: illustrated in Figure 10.
- $bool \leftarrow \mathsf{PMBTB.Verify}((\mathbf{x}, \mathbf{y}), t, \sigma)$: return **true**.
- $\mathsf{ind} \leftarrow \mathsf{PMBTB.ReadBit}((\mathbf{x}, \mathbf{y}), t, \sigma)$: read σ as $(S_0, S_1, W_0, W_1) \in \mathbb{G}^4$. Then,
  (a) 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;
  (b) 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;
  (c) else, return $\perp$.

```
PMBTB.User($(\mathbf{X}, \pi), t$)                                  PMBTB.Sign($(\mathbf{x}, \mathbf{y}), b$)

  PMBTB.User$_0$($(\mathbf{X}, \pi), t$)
  // Check once: $\Pi_{\mathsf{DLOGAND2}}.\mathsf{Verify}(\mathbf{X}, \pi)$
  $T := \mathsf{H}_t(t)$
  for $d = 0, 1$ :
      $r_d, \rho_d \leftarrow\$ \mathbb{Z}_p^*$
      $T'_d := r_d^{-1} \cdot (T - \rho_d G)$
  return $\big((T'_0, T'_1), ((r_d, \rho_d, T'_d)_{d \in \{0,1\}}, t)\big)$

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

                                              PMBTB.Sign$_0$($(\mathbf{x}, \mathbf{y}, b), (T'_0, T'_1)$)
                                              $s \leftarrow\$ \{0,1\}^\lambda$
                                              $S'_b := \mathsf{H}_s(T'_b, s)$
                                              $W' := x_b T'_b + y_b S'_b$
                                              return $(s, W')$

                          $\xleftarrow{s, W'}$

  PMBTB.User$_1$($\big((r_d, \rho_d, T'_d)_{d \in [2]}, t\big), (s, W')$)
  for $d = 0, 1$ :
      $S_d := r_d \mathsf{H}_s(T'_d, s) + \rho_d H$
      $W_d := r_d W' + \rho_d X_d$
  return $\sigma := (S_0, S_1, W_0, W_1)$
```

Fig. 10. Token issuance for PMBTB (Construction 5).

Correctness. Honestly-generated tokens are always valid, because the verification algorithm PMBTB.Verify always outputs **true**. We thus focus on Equation (7). By correctness of the underlying proof system, the protocol aborts only with negligible probability. If the user returns a tuple $(S_0, S_1, W_0, W_1) \in \mathbb{G}^4$, then there exists $b \in \{0, 1\}$ such that:

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

The probability that the above holds for both $b = 0$ and $b = 1$ (in which case, PMBTB.ReadBit returns $\perp$) is statistically negligible. In fact, if:

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

we have two possible cases:

(a) $x_0 = x_1$, which in turn implies that:

$$S_0 = \frac{y_1}{y_0} S_1.$$

However, because S_0 is distributed uniformly at random in $\mathbb{G}$, this happens with probability $1/p$.

(b) $x_0 \neq x_1$, which in turn implies that:

$$\mathsf{H}_t(t) = \frac{1}{x_1 - x_0}(y_0 S_0 - y_1 S_1),$$

which happens with probability $1/p$.

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

Theorem 13. *If CTGDH holds for the group generator algorithm* GrGen *and* Π_{DLOGAND2} *is a zero-knowledge proof system for* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PMBTB},\ell}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda).$$

Inuitively, unforgeability follows a similar reasoning of Theorem 8 (unforgeability of PMBT) except that here, at issuance time, we are sending strictly less information to the user, since we removed the NIZK at issuance time. The proof of knowledge of the discrete log, published at the beginning within the public parameters, can be simulated by zero knowledge.

Theorem 14. *If DDH holds for the group generator* GrGen *and* Π_{DLOGAND2} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *is 3-unlinkabile.*

A PPT adversary in the game $\mathrm{UNLINK}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)$ can now: compute W' using (x_0, y_0), using (x_1, y_1), or yet another key. Specifically in the latter case, the token σ will be distributed at random independently from W'. In the full version, we prove that now the adversary can partition the set of open sessions at most in 3, and that therefore the advantage in the game $\mathrm{UNLINK}_{\mathsf{PMBTB},\mathsf{A}}(\lambda) \leq 3/m + \mathsf{negl}(\lambda)$.

Theorem 15. *If DDH holds for the group generator* GrGen *and* Π_{DLOGAND2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *provides private metadata bit with advantage:*

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBTB}}(\lambda) \leq \frac{O(q^2)}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 4\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda),$$

where q is the number of queries the adversary makes either to H_s *or* $\textsc{Sign}$*.*

As for unforgeability, the proof follows a similar reasoning to the case of PMBT. We notice that now the issuer is sending strictly less information during signing, and that the zero-knowledge proof π can be simulated.

8 Implementation

We implemented our construction in pure Rust (stable, version `1.41.0`), using the Ristretto group[10] on the top of Curve25519 [Ber06], as provided by `curve25519-dalek`[11]. The second generator

[10] https://datatracker.ietf.org/doc/draft-hdevalence-cfrg-ristretto/
[11] https://dalek.rs/

Table 2. Benchmarks for our constructions.

Constructions	DLEQ/DLEQOR		User		Issuer		
	Prove	Verify	Token Gen.	Unblinding	Key Gen.	Signing	Redemption
PP [DGS+18]	212 µs	181 µs	111 µs	286 µs	84 µs	303 µs	95 µs
PMBT	576 µs	666 µs	135 µs	844 µs	234 µs	845 µs	235 µs
PPB	–	–	197 µs	164 µs	190 µs	87 µs	95 µs
PMBTB	–	–	368 µs	678 µs	512 µs	155 µs	247 µs

$H \in \mathbb{G}$ is chosen by hashing into the group the public generator G. Hashing into the group is done with a Elligator 2 map [Tib14] with SHA-512. Using `rust-wasm`[12], 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.[13]

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. As expected, Constructions 4 and 5 feature very fast issuance time at a slight increase in the user computation. Our results are between ten and one thousand faster than the previous implementation proposed in [DGS+18] due to the different choice[14] of elliptic curve (NIST P-256) as well as the programming language used.

Acknowledgments

The authors thank Fabrice Benhamouda, Dan Boneh, Charlie Harrison, Michael Kleber, Hugo Krawczyk, Steven Valdez, and Moti Yung for helpful discussions. We thank Serge Vaudenay for pointing out minor errors and typos in the CMBT construction in the revisions of 2021-01-13, compared to the original version. A part this work was completed while Michele Orrù was an intern at Google and at Web3 Foundation.

References

ABR01. Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In *CT-RSA*, volume 2020 of *LNCS*, pages 143–158. Springer, 2001.

ANN06. Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In *CT-RSA 2006*, 2006.

Ber06. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. pages 207–228, 2006.

Ber14. David Bernhard. *Zero-knowledge proofs in theory and practice.* PhD thesis, University of Bristol, 2014.

BFPV11. 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.

BG04. Daniel R. L. Brown and Robert P. Gallant. The static diffie-hellman problem. *IACR Cryptology ePrint Archive*, 2004:306, 2004.

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

[12] `https://rustwasm.github.io/`
[13] `https://github.com/mmaker/anonymous-tokens`
[14] We comment on the choice of curve and its security in Appendix C.

BMW03. 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.

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

Bol03. 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.

BP02. 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.

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In *SCN*, volume 7485 of *LNCS*, pages 95–112. Springer, 2012.

CA89. David Chaum and Hans Van Antwerpen. Undeniable signatures. In *CRYPTO*, volume 435 of *LNCS*, pages 212–216. Springer, 1989.

Cam97. Jan Camenisch. Efficient and generalized group signatures. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 465–479. Springer, 1997.

CDS94. 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.

Cha82. David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, 1982.

Cha90. David Chaum. Zero-knowledge undeniable signatures. In *EUROCRYPT*, volume 473 of *LNCS*, pages 458–464. Springer, 1990.

Che06. Jung Hee Cheon. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

CHM$^+$19. 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.

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

CL01. 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.

CMZ14. Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. pages 1205–1216, 2014.

CP92. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, volume 740 of *LNCS*, pages 89–105. Springer, 1992.

CvH91. David Chaum and Eugène van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, 1991.

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

ECS$^+$15. 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.

FHKS16. 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.

FHS15. 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.

Fis06. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, volume 4117 of *LNCS*, pages 60–77. Springer, 2006.

FPS19. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures in the algebraic group model. Cryptology ePrint Archive, Report 2019/877, 2019.

GG14. Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 477–495. Springer, 2014.

Gha13. Essam Ghadafi. Formalizing group blind signatures and practical constructions without random oracles. In *ACISP*, volume 7959 of *LNCS*, pages 330–346. Springer, 2013.

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

HL06. Javier Herranz and Fabien Laguillaumie. Blind ring signatures secure under the chosen-target-cdh assumption. In *ISC*, volume 4176 of *LNCS*, pages 117–130. Springer, 2006.

JKK14. 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.

JKR18. Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Threshold partially-oblivious prfs with applications to key management. *IACR Cryptology ePrint Archive*, 2018:733, 2018.

JKX18. 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.

JL10. Stanislaw Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In *SCN*, volume 6280 of *LNCS*, pages 418–435. Springer, 2010.

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

LQ04. Benoît Libert and Jean-Jacques Quisquater. The exact security of an identity based signature and its applications. *IACR Cryptology ePrint Archive*, 2004:102, 2004.

LR98. Anna Lysyanskaya and Zulfikar Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography*, 1998.

Oka92. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, volume 740 of *LNCS*, pages 31–53. Springer, 1992.

OP01. 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.

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

Ram13. 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.

SC12. 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.

Sch01. 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.

Sch06. Claus Peter Schnorr. Enhancing the security of perfect blind dl-signatures. *Information Sciences*, 176(10):1305 – 1320, 2006.

TAKS07. 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.

Tib14. 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.

TPY^{+}19. 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.

A Security assumptions

We recall here the classical discrete logarithm, decisional Diffie–Hellman, and computational Diffie–Hellman assumptions that we will use throughout the paper. Then, we discuss the chosen-target Diffie-Hellman, and justify our choice of introducing its gap variant, CTGDH, for assessing the security of the anonymous tokens proposed.

A.1 DL, DDH, CDH

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 $X = xG$. More formally, we say that DLOG holds for the group generator GrGen if for any PPT adversary A:

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

where $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Figure 11.

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 PPT adversary has negligible advantage in distinguishing $C \leftarrow_{\$} \mathbb{G}$ from the Diffie–Hellman $C = abP$. More formally, we say that DDH holds for the group generator GrGen if for any PPT adversary A:

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

is negligible in λ, where $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Figure 11.

Computational Diffie–Hellman. The computational Diffie–Hellman (CDH) 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 PPT adversary has negligible advantage in returning $C = abP$. More formally, we say that CDH holds for the group generator GrGen if for any PPT adversary A:

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

where $\mathrm{CDH}_{\mathsf{GrGen},\mathsf{A}}(\lambda)$ is defined in Figure 11.

A.2 Chosen-target Diffie–Hellman

The CTDH assumption [Bol03] has proven itself very useful for a range of applications: password-authenticated key exchanges [JKX18], signatures [LQ04], and private-set intersection [JL10]. One can also find it called in the literature as *one-more Diffie–Hellman* (OMDH) assumption [BP02, BNPS03], both in the "chosen-target" flavor (where the adversary can chose the $\ell + 1$ subset of challenges to solve) and the "known-target" flavor (where the adversary must solve a fixed set of $\ell + 1$ challenges). Known-target one-more Diffie–Hellman is equivalent to chosen-target Diffie–Hellman [KM08].

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

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

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

Fig. 11. The games for discrete logarithm, decisional Diffie–Hellman, and computational Diffie–Hellman.

```
Game $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$
$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathbb{Z}_p;\quad X := xG$
$q := 0;\quad \mathcal{Q} = [\,]$
$(t_i, Z_i)_{i \in [\ell+1]} \leftarrow \mathsf{A}^{\mathrm{TARGET},\mathrm{HELP}}(\Gamma, X)$
for $i \in [\ell + 1]$ :
  if $t_i \notin \mathcal{Q}$ then return $0$
  $Y_i := \mathcal{Q}[t_i]$
return $\big(q < \ell$ and
            $\forall i \neq j \in [\ell + 1]\ \ t_i \neq t_j$ and
            $\forall i \in [\ell + 1]\ \ x \cdot Y_i = Z_i\big)$
```

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

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

Fig. 12. The Chosen-Target Diffie–Hellman assumption.

Definition 16 (Chosen-target Diffie–Hellman). *The chosen-target gap Diffie–Hellman assumption for the group generator* GrGen *states that for any* PPT *adversary* A*, and any* ℓ*,* A *has negligible advantage in solving CDH on* $\ell + 1$ *target group elements, even if* A *is given access to a CDH oracle for* ℓ *instances. More formally, for any* PPT *adversary* A*, any* $\ell > 0$*:*

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

where $\mathrm{CTDH}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$ *is defined in Figure 12.*

A.3 Equivalence of CTDH and OMD

Privacy Pass, the construction introduced in [DGS+18], 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 $c \leftarrow\!\!\$\ \mathbb{Z}_p$.

```
Game $\mathrm{OMD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$
$\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
$(x, X) \leftarrow \mathsf{Elg.KeyGen}(\Gamma)$
for $i \in [\ell+1]$ :
    $M_i \leftarrow\!\!\$\ \mathbb{G}$
    $c_i \leftarrow \mathsf{Elg.Enc}(X, M_i)$
// $\textsc{Dec}(c^*) = \mathsf{Elg.Dec}(x, c^*)$, at most $\ell$ times
$(M_i^*)_{i\in[\ell+1]} = \mathsf{A}^{\textsc{Dec}}(X, c_0, \ldots, c_\ell)$
return $(\forall i \in [\ell+1] :\ M_i^* = M_i)$
```

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

Fig. 13. Game for one-more decryption (left) and reduction to chosen-target Diffie–Hellman (right).

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 17. *CTDH holds for* GrGen *if and only if* Elg[GrGen] *has one-more decryption security. More precisely, for all $\ell > 0$:*

$$\mathsf{Adv}^{\mathrm{ctdh}}_{\mathsf{GrGen},\ell}(\lambda) = \mathsf{Adv}^{\mathrm{omd}}_{\mathsf{GrGen},\ell}(\lambda).$$

Proof. Let $\ell \in \mathbb{N}$. We start by proving that OMD $\Longrightarrow$ CTDH. Let A be a PPT adversary for $\mathrm{OMD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$. 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 by querying the 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 $\mathrm{OMD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$ (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 $\mathrm{OMD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$, 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 $\mathrm{OMD}_{\mathsf{Elg[GrGen]},\mathsf{A},\ell}(\lambda)$, therefore for all PPT adversaries A:

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

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

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 $c_0, \ldots, c_\ell$ 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:

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

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

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

□

A.4 The gap variant of CTDH

Recall that we formalize the notion of *chosen-target* *gap* *Diffie–Hellman* (CTGDH) problem in Section 2.1, the gap problem equivalent to the CTDH problem. Indeed, the chosen-target DH problem was originally introduced by Boldyreva [Bol03] in *Gap DH* groups [BLS01], 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 Appendix A.2 did not ask for the group $\mathbb{G}$ to be a Gap DH group.

This formalization is not merely an artifact of the proof. Indeed, when a user is redeeming anonymous tokens, it is likely that her 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 if a token of their choice satisfies the verification equation. This specific behavior is not unique to our anonymous tokens primitive. In particular, a similar issue arised many times in the literature: when proving the CCA security of the (hashed) El Gamal encryption scheme [ABR01, CKS09], the unforgeability of Chaum's undeniable signature scheme [CA89, Cha90, OP01], unforgeability of blind signatures [BLS01], and UC-security of the *2Hash-DH* VOPRF [JKK14]. All these schemes are proved under a *gap* problem [OP01], i.e., a computational problem that gives oracle access to the underlying decision problem.[15] For example, [OP01] defines the Gap DH problem, which given a triple (P, aP, bP), ask to find the element abP with the help of a Decision Diffie–Hellman oracle (which answers whether (X, Y, Z) is a valid DH triple).

B NIZK instantiations

Throughout this work, we assumed the existence of a non-interactive, knowledge-sound and zero-knowledge proof system. We described our anonymous tokens protocols in a modular way, so that any knowledge-sound and zero-knowledge proof system can be used (e.g. SNARKs, Groth-Sahai proofs, Σ-protocols). Here, we propose a simple and efficient instantiation of them in the random oracle model. Equations (1) to (5) define proof systems for the following relations:

$$\mathsf{R}_{\mathsf{DLOG}} \coloneqq \{(X, x) \in \mathbb{G} \times \mathbb{Z}_p : \ X = xG\}; \tag{10}$$

$$\mathsf{R}_{\mathsf{DLOGAND2}} \coloneqq \left\{(\mathbf{X}, (\mathbf{x}, \mathbf{y})) \in \mathbb{G}^2 \times (\mathbb{Z}_p^2 \times \mathbb{Z}_p^2) : \ \forall i \in \{0, 1\} \ \ X_i = x_i G + y_i H\right\}; \tag{11}$$

$$\mathsf{R}_{\mathsf{DLEQ}} \coloneqq \left\{ ((X, T, W), x) \in \mathbb{G}^3 \times \mathbb{Z}_p : \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} \right\}; \tag{12}$$

$$\mathsf{R}_{\mathsf{DLEQ2}} \coloneqq \left\{ ((X, T, S, W), (x, y)) \in \mathbb{G}^4 \times \mathbb{Z}_p^2 : \ \begin{bmatrix} X \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \right\}; \tag{13}$$

$$\mathsf{R}_{\mathsf{DLEQOR2}} \coloneqq \left\{ \begin{array}{c} ((\mathbf{X}, T, S, W), (b, x, y)) \in \mathbb{G}^5 \times ([2] \times \mathbb{Z}_p \times \mathbb{Z}_p) : \\ \begin{bmatrix} X_b \\ W \end{bmatrix} = x \begin{bmatrix} G \\ T \end{bmatrix} + y \begin{bmatrix} H \\ S \end{bmatrix} \end{array} \right\}. \tag{14}$$

All above relations are (implicitly) parameterized in the group description Γ, and can be seen as facilities of relationships indexed by $\Gamma \in [\mathsf{GrGen}(1^\lambda)]$.

Σ-protocols are interactive, special sound, and special honest-verifier zero-knowledge. They be turned into non-interactive zero-knowledge arguments using the Fiat-Shamir transform in the random oracle model. In Bernhard's PhD thesis [Ber14] it is claimed that the Fiat–Shamir transformation of a Σ-protocol is simulation extractable, a soundness property that even stronger than knowledge soundness.

Theorem 18 ([Ber14, Thm. 5.14]). *Let Σ be a Σ-protocol. If the challenge space is exponential, and the verification equation is a group morphism on the response, the strong Fiat-Shamir transformation $\mathsf{FS}[\Sigma]$ of Σ is simulation-sound extractable.*

In Figure 14 we present a protocol for relation $\mathsf{R}_{\mathsf{DLEQOR2}}$ (Eq. (14)), which can be easily adapted to be a proof system also for relations (10) to (12). The setup algorithm generates the group description $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$, The proving algorithm (Fig. 14, left) proceeds simulating the transcripts for all $i \in [2], i \neq b$, 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 $\mathsf{H}_c(\mathbf{X}, T, S, W, \mathbf{K}_0, \mathbf{K}_1)$.

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

```
$\Sigma_{\mathsf{DLEQOR2}}.\mathsf{Prove}((\mathbf{X}, T, S, W), (b, x_b, y_b))$
$(k_0, k_1) \leftarrow\!\!\$\, \mathbb{Z}_p^2$
$\mathbf{K}_b := k_0 \cdot (G; T) + k_1 \cdot (H; S)$
for $i \in [n], \quad i \neq b$
    $c_i, u_i, v_i \leftarrow\!\!\$\, \mathbb{Z}_p$
    $\mathbf{K}_i := u_i \cdot (G; T) + v_i \cdot (H; S) - c_i \cdot (X_i; W)$
$c := \mathsf{H}_c(\mathit{\Gamma}, (\mathbf{X}, T, S, W), \mathbf{K}_0, \ldots, \mathbf{K}_n)$
$c_b := c - \sum_{i \neq b} c_i$
$u_b := k_0 + c_b x_b$
$v_b := k_1 + c_b y_b$
return $(\mathbf{c}, \mathbf{u}, \mathbf{v})$
```

```
$\Sigma_{\mathsf{DLEQOR2}}.\mathsf{Verify}((\mathbf{X}, S, T, W), (\mathbf{c}, \mathbf{u}, \mathbf{v}))$
$\forall i \in [n]: \quad \mathbf{K}_i := u_i(G; T) + v_i \cdot (H; S) - c_i(X_i; W)$
$c := \sum_i c_i$
return $c = \mathsf{H}_c(\mathit{\Gamma}, (\mathbf{X}, T, S, W), \mathbf{K}_0, \ldots, \mathbf{K}_n)$
```

Fig. 14. The protocol $\Sigma_{\mathsf{DLEQOR2}}$.

The protocol has special soundness by standard OR-composition of sigma protocols [CDS94]: from two transcripts $(\mathbf{c}, \mathbf{u}, \mathbf{v})$ and $(\mathbf{c}', \mathbf{u}', \mathbf{v}')$ verifying simultaneously, some $b \in [2]$ such that $c_b \neq c_b'$ it is possible to extract a witness (x_b, y_b) by computing $x_b := (u_b - u_b')/(c_b - c_b')$ and $y_b := (v_b - v_b')/(c_b - c_b')$.

The protocol is also zero-knowledge: the simulator simply produces valid transcripts for all $i \in [2]$ by selecting $c_i, u_i, v_i \leftarrow\!\!\$\, \mathbb{Z}_p$, and computing $\mathbf{K}_i := u_i(G; T) + v_i(H; S) - c_i(X_i, W)$. Then, it computes $c := \sum_i^n c_i$ and programs the random oracle to reply with c when queried on $(\mathit{\Gamma}, \mathbf{X}, T, S, W, \mathbf{K}_0, \mathbf{K}_1)$. 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.

B.1 Batching proofs

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

Theorem 19. *If DLOG holds for the group generator* GrGen*, then the proof system* $\Sigma_{\mathsf{DLEQOR2}}[\mathsf{GrGen}]$ *is a zero-knowledge argument of knowledge for relation* $\mathsf{R}_{\mathsf{DLEQOR2}}$.

Proof (Knowledge soundness). Knowledge soundness for the batched protocol follows from knowledge soundness of the underlying $\Sigma_{\mathsf{DLEQOR2}}$ proof system, plus a negligible statistical error. By soundness of the underlying proof system we have that, except with a negligible extraction error, if $\Sigma^{\mathsf{batched}}_{\mathsf{DLEQOR2}}.\mathsf{Verify}(\mathit{\Gamma}, \mathbf{X}, \overline{T}, \overline{S}, \overline{W}, \pi) = \mathbf{true}$, then it is possible to extract (b, x, y) such that $\big((\mathbf{X}, \overline{T}, \overline{S}, \overline{W}), (b, x, y)\big) \in \mathsf{R}_{\mathsf{DLEQOR2}}$. 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 $(\mathbf{X}, T_i, S_i, W_i)$, 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

```
$\Sigma^{\text{batched}}_{\mathsf{DLEQOR2}}.\mathsf{Prove}((\mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W}), (x, y))$
$e_0, \ldots, e_m \coloneqq \mathsf{H}_e(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W})$
$\overline{T} \coloneqq \sum_j^m e_j T_j$
$\overline{S} \coloneqq \sum_j^m e_j S_j$
$\overline{W} \coloneqq \sum_m^m e_j W_j$
return $\mathsf{DLEQOR}.\mathsf{Prove}((\mathbf{X}, \overline{T}, \overline{S}, \overline{W}), (x, y))$
```

```
$\Sigma^{\text{batched}}_{\mathsf{DLEQOR2}}.\mathsf{Verify}((\mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W}), \pi)$
$e_0, \ldots, e_m \coloneqq \mathsf{H}_e(\Gamma, \mathbf{X}, \mathbf{T}, \mathbf{S}, \mathbf{W})$
$\overline{T} \coloneqq \sum_j^m e_j T_j$
$\overline{S} \coloneqq \sum_j^m e_j S_j$
$\overline{W} \coloneqq \sum_m^m e_j W_j$
return $\mathsf{DLEQOR}.\mathsf{Verify}((\mathbf{X}, \overline{T}, \overline{S}, \overline{W}), \pi)$
```

Fig. 15. Batching Π_{DLEQOR2}.

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:

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

□

C 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 [BG04] and Cheon's attack [Che06], as already studied independently by Taylor Campbell[16] in the context of Privacy Pass, by Thomas et al. [TPY+19] in the context of OPRFs, and by Chiesa et al. [CHM+19] 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 [TPY+19]. In the case of Ristretto, $p-1$ is not smooth. $p-1$ 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

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

```
Game $\mathrm{HIJACK}_{\mathsf{AT},\mathsf{A}}(\lambda)$
$\mathcal{Q} \coloneqq [\,]$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{AT.Setup}(1^\lambda)$
$(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(\mathsf{crs})$
$(t, R, \bar{\sigma}) \leftarrow \mathsf{A}^{\textsc{Token}}(\mathsf{pp})$
return $(t, R) \notin \mathcal{Q}$ and $\mathsf{AT.Bind}(\mathsf{crs}, \mathsf{sk}, t, R, \bar{\sigma})$

Oracle $\textsc{Token}(b, t, R)$
$\sigma \leftarrow \langle \mathsf{AT.User}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b) \rangle$
$\bar{\sigma} \coloneqq \mathsf{AT.Spend}(\mathsf{pp}, t, R, \sigma)$
append $(t, R)$ to $\mathcal{Q}$
return $\bar{\sigma}$
```

Fig. 16. Token hijacking game.

$O(2^{94})$ 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.

D Token Hijacking

Let us now consider a stronger attacker, a man-in-the middle capable of intercepting messages between the user and the issuer at redemption time. Despite such an attacker cannot forge or de-anonymize an user, it is capable of hijacking user tokens sent for redemption, and spend them on another resource than initially destined for. This attack model is not covered in our definitions, since we assume that the communication between the user and the issuer always to happen over a secure channel. We discuss in this section how it is possible to cover also the above scenario.

We model attack vector of a token hijacker as a security experiment where the adversary interacts with with an oracle and request request a token for a specified resource identifier $R \in \{0,1\}^*$ (e.g. a), a private metadata bit $b \in [2]$, and randomness $t \in \{0,1\}^\lambda$. After interacting an arbitrary number of times with this oracle, the adversary must return a valid token $(t, \bar{\sigma})$ bound to a resource R that was never queried by the oracle. More formally, we introduce two new procedures: $\mathsf{AT.Bind}$, that cryptographically binds a token with a specified resource, and $\mathsf{AT.Spend}$ that allows to verify (spend) a token on the selected resource. More formally, we say that an anonymous token AT is secure against hijacking if, for all PPT adversaries A:

$$\mathsf{Adv}^{\text{hijack}}_{\mathsf{AT},\mathsf{A}}(\lambda) \coloneqq \left|\Pr\left[\mathrm{HIJACK}_{\mathsf{AT},\mathsf{A}}(\lambda)\right]\right| = \mathsf{negl}(\lambda)\,.$$

Informally, this can be achieved with the help of a MAC: the user hashes W into a MAC key $\mathsf{H}_k(W)$ and signs the resource R with this new key. On the other hand, the issuer uses the randomness t (along with any additional information, such as S) and the secret key x to compute the same shared key $\mathsf{H}_k(W)$. More concretely, in the case of Privacy Pass (Section 4) we set $\overline{\mathsf{PP}}.\mathsf{Bind}(\mathsf{pp}, t, R, \sigma) \coloneqq \mathsf{MAC.Sign}(\mathsf{H}_k(\sigma), t, R)$ and $\overline{\mathsf{PP}}.\mathsf{Spend}(\mathsf{crs}, \mathsf{sk}, t, R, \bar{\sigma}) \coloneqq \mathsf{MAC.Ver}(\mathsf{H}_k(x\mathsf{H}_t(t)), t, R)$.

Theorem 20. *If* MAC *is an existentially-unforgeable message authentication code, and* PP *is a one-more unforgeable anonymous token, then* $\overline{\mathsf{PP}}$ *prevents token hijacking.*

Proof. The proof proceeds by means of a hybrid argument:

Hyb_0 This is the game $\mathrm{HIJACK}_{\mathsf{PP},\mathsf{A}}(\lambda)$. The adversary wins if it manages to redeem a token $\bar{\sigma}$ for a resource R that was not previously redeemed. That is, the adversary wins if it outputs a tuple $(R, t, \bar{\sigma})$ such that $\mathsf{MAC.Ver}(k, t, R) = \textbf{true}$ with $k = \mathsf{H}_k(x \cdot \mathsf{H}_t(t))$ and $(t, R) \notin \mathcal{Q}$.

Hyb_1 In this hybrid, we abort if the adversary never made a query to the random oracle H_k of the form $W = x \cdot \mathsf{H}_t(t)$ during its execution. For any PPT adversary A making at most $q(\lambda) = \mathsf{poly}(\lambda)$ queries to the random oracle H_k, this hybrid is indistinguishable from the previous one. Therefore:

$$\frac{q(\lambda)}{p} \leq \left|\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PP},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PP},\mathsf{A}}(\lambda)\right|.$$

Hyb_2 We retrieve the query $W \in \mathbb{G}$ associated to the MAC key $k = \mathsf{H}_k(W)$ of the forgery, and we run the verification equation of the anonymous token $\mathsf{PP.Ver}(\mathsf{p}, t, W)$. If it returns **false**, we terminate the game immediately and the adversary wins.
If the output of a PPT adversary A in the hybrid Hyb_1 is distinguishable from the one in Hyb_2, then it is possible to build an adversary B that wins the game $\mathrm{UNF}_{\mathsf{MAC},\mathsf{B}}(\lambda)$ with non-negligible probability.

At this point, the adversary wins only if it outputs a token $(t, \bar{\sigma})$ for a resource R and there exists a query to the random oracle H_k for $W \in \mathbb{G}$ such that (t, W) was not previously produced by the oracle TOKEN. This is exactly the one-more unforgeability experiment for PP. It follows that:

$$\mathsf{Adv}^{\mathrm{hijack}}_{\mathsf{PP},\mathsf{A}}(\lambda) \leq \frac{q(\lambda)}{p} + \mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PP},\mathsf{A}}(\lambda).$$

□

E Security proofs for Privacy Pass (PP)

E.1 Unforgeability

The one-more unforgeability security notion from Privacy Pass [DGS+18] did not provide the adversary with a VERIFY oracle. One-more unforgeability without verification oracle is proven under the one-more decryption of El Gamal problem, which we prove to be equivalent to the chosen-target Diffie–Hellman problem in Appendix A.3. We therefore prove the following corollary:

Theorem 4. *If CTGDH holds for* GrGen *and* Π_{DLEQ} *is a zero-knowledge proof system for relation* R_{DLEQ}*, then* $\mathsf{PP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PP},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ}},\mathsf{R}_{DLEQ}}(\lambda).$$

Proof (sketch). The proof follows directly from the proof of Theorem 6. We detail highlight here the differences:

- In the hybrid, the proof system is with respect to the proof system Π_{DLEQ} instead of Π_{DLEQ2}.
- In the reduction to chosen-target gap Diffie–Hellman, B sets the public parameters as $A \in \mathbb{G}$, and answers the signing queries with whatever HELP answers and the simulated proof. At the end of the game, it outputs whatever A outputs.

It follows that that:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PP},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ}},\mathsf{R}_{\mathsf{DLEQ}}}(\lambda).$$

□

E.2 Unlinkability

In [DGS+18], the Privacy Pass protocol is proved to be unconditionally unlinkable (up to the soundness error of the proof system).

Theorem 5. *Let* GrGen *be a group generator algorithm. If* Π_{DLEQ} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathit{DLEQ}}$*, then* $\mathsf{PP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ}}]$ *is 1-unlinkable.*

Proof. Consider an hybrid game Hyb in which we use the knowledge extractor to recover the (unique) witness x, and always compute W as $xT = x\mathsf{H}_t(t)$. This hybrid is indistinguishable from the original game by the knowledge soundness of the proof system:

$$q_0 \cdot \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLEQ}},\mathsf{R}_{\mathsf{DLEQ}},\mathsf{B},\mathsf{Ext}}(\lambda) \geq \left| \mathsf{Adv}^{\mathrm{hyb}}_{\mathsf{PP},\mathsf{A}}(\lambda) - \Pr\left[\mathrm{UNLINK}_{\mathsf{PP},\mathsf{A}}(\lambda) = 1\right] \right|,$$

where q_0 is the total number of calls to $\mathsf{PP}.\mathsf{User}_1$.

Now, consider a second hybrid in which we program the random oracle so that T' is drawn uniformly at random and $\mathsf{H}_t(t) := rT'$. The distribution is unchanged, and it holds that T' and (t, W) are independent. Therefore

$$\Pr\left[\mathrm{UNLINK}_{\mathsf{PP},\mathsf{A}}(\lambda) = 1\right] \leq \frac{1}{m} + q_0 \cdot \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLEQ}},\mathsf{R}_{\mathsf{DLEQ}},\mathsf{B},\mathsf{Ext}}(\lambda)$$

which concludes the proof. □

F Security proofs for OSPP

F.1 Unforgeability

Theorem 6. *If CTGDH holds for* GrGen*, and* Π_{DLEQ2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathit{DLEQ2}}$*, then* $\mathsf{OSPP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{OSPP},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathit{DLEQ2}}}(\lambda).$$

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

Hyb_0 This is the game $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$. The adversary is provided with $X \in \mathbb{G}$, and has access to the signing oracle SIGN, the verification oracle VERIFY, and the random oracles H_s (used for the response of the issuer), H_t (used for blinding the message by the user). At the end of its execution, it must output $\ell + 1$ valid tokens.

Hyb_1 This hybrid replaces the way zero-knowledge proofs are generated when answering signing oracles: instead of using the proving algorithm $\Pi_{\mathsf{DLEQ2}}.\mathsf{Prove}$, we use the zero-knowledge simulator $\Pi_{\mathsf{DLEQ2}}.\mathsf{Sim}$ for all signing queries. If there exists a PPT adversary A whose advantage is different between the two games, then it is possible to construct an adversary for zero-knowledge of the underlying proof system. Consider the following PPT adversary B for the game $\mathrm{ZK}^{\beta}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}},\mathsf{B}}(\lambda)$. B takes as input $(\mathit{\Gamma}, \mathsf{pcrs})$ and generates the public parameter X following $\mathsf{OSPP}.\mathsf{KeyGen}$. It then invokes the adversary A on X, and for any signing query, it generates the proofs π's the PROVE oracle. 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 PPT adversary A, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}},\mathsf{B}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{AT},\mathsf{A},\ell}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)\right| .$$

At this point, we prove that if there is a PPT adversary A that has non-negligible advantage $\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$, then we can construct a PPT adversary B that has non-negligible advantage in the chosen-target gap Diffie–Hellman game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$. B receives as input the group description Γ and a challenge $A \in \mathbb{G}$ as input. It runs $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQ2}}.\mathsf{Setup}(\Gamma)$, then samples $y \leftarrow \mathbb{Z}_p^*$, and computes $X := A + yH$. Then, it invokes the adversary $\mathsf{A}(\mathsf{crs}, X)$, where $\mathsf{crs} := (\Gamma, \mathsf{pcrs})$. Note that X is distributed as in Hyb_1. The adversary B responds to the queries that A makes to the oracles H_t, $\textsc{Sign}$, and $\textsc{Verify}$ 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 to the oracle $\textsc{Sign}(T')$, B samples $s \leftarrow\!\!\$\ \{0,1\}^\lambda$ and defines $S' := \mathsf{H}_s(T', s)$. Then it invokes the oracle $Z := \textsc{Help}(T')$, defines $W' = Z + yS'$, and simulates the proof π. Finally, it returns (s, W', π);
– to any query to the oracle $\textsc{Verify}(t, (S, W))$, B sets $T := \mathsf{H}_t(t)$ and invokes the oracle $\textsc{Ddh}(T, W - yS)$, and returns whatever it returns.

(The random oracle H_s is left unchanged.) First, note that the distributions of H_t, $\textsc{Sign}$, and $\textsc{Verify}$ are identical to the ones of Hyb_1. 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 $(t_i, (W_i - yS_i))_{i\in[\ell+1]}$.

We claim that B wins the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ every time that A wins Hyb_1: by the winning condition in Hyb_1, A won if and only if all t_i are different and $W_i = x\textsc{Target}(t_i) + yS_i$ where x is the unique element of $\mathbb{Z}_p$ such that $A = xG$. Furthermore, by the winning condition, A only called the signing oracle $\textsc{Sign}$ at most ℓ times (and thus $\textsc{Help}$ was called at most ℓ times). Therefore:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{OSPP},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}}}(\lambda).$$

and this concludes the proof. □

F.2 Unlinkability

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

Theorem 7. *If DDH holds for* GrGen *and* Π_{DLEQ2} *is an argument of knowledge for relation* $\mathsf{R}_{\mathsf{DLEQ2}}$*, then* $\mathsf{OSPP}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}]$ *is 1-unlinkable.*

Proof. The theorem trivially holds for $m = 1$. Let $m > 1$. We prove the theorem via a sequence of hybrids, summarized in Figure 17.

Hyb_0 This hybrid is the game $\mathrm{UNLINK}_{\mathsf{OSPP},\mathsf{A},m}(\lambda)$, where the adversary is given as a challenge the j-th token.

```
Game $\mathrm{UNLINK}_{\mathsf{OSPP},\mathsf{A},m}(\lambda)$ in Hyb0, Hyb2, Hyb3, Hyb5

$\Gamma := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQ2}}.\mathsf{Setup}(\Gamma)$
$(\mathsf{st}, X) \leftarrow \mathsf{A}((\Gamma, \mathsf{pcrs}))$
$q_0 := 0;\ q_1 := 0;\ \mathcal{Q} := \emptyset$
$(\mathsf{st}, (s_i, W'_i, \pi_i)_{i\in\mathcal{Q}}) \leftarrow \mathsf{A}^{\mathrm{USER}_0,\mathrm{USER}_1}(\mathsf{st})$
if $\mathcal{Q} = \emptyset$ then return $0$
$j \leftarrow_\$ \mathcal{Q};\ \ \mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$\sigma_j \leftarrow \mathsf{OSPP}.\mathsf{User}_1(\mathsf{st}_j, (s_j, W'_j, \pi_j))$
if $\mathcal{Q} = \emptyset$ then return $0$
$k \leftarrow_\$ \mathcal{Q};\ \ \mathcal{Q} := \mathcal{Q} \setminus \{k\}$
$\sigma_k \leftarrow \mathsf{OSPP}.\mathsf{User}_1(\mathsf{st}_k, (s_k, W'_k, \pi_k))$
for $i \in \mathcal{Q} : \sigma_i \leftarrow \mathsf{OSPP}.\mathsf{User}_1(\mathsf{st}_i, (s_i, W'_i, \pi_i))$
  if not $(x_i, y_i) = (x_k, y_k)$ then abort
  $W_j := x_k \mathsf{H}_t(t_j) + y_k S_j$
  if $\Pi_{\mathsf{DLEQ2}}.\mathsf{Verify}(\mathsf{pcrs}, (X, T'_j, S'_j, W'_j), \pi_j)$ then
      $\sigma_j := (t_j, (S_j, W_j))$
  $W_k := x_k \mathsf{H}_t(t_k) + y_k S_k$
  if $\Pi_{\mathsf{DLEQ2}}.\mathsf{Verify}(\mathsf{pcrs}, (X, T'_k, S'_k, W'_k), \pi_k)$ then
      $\sigma_k := (t_k, (S_k, W_k))$
$\phi \leftarrow \mathcal{S}_\mathcal{Q}$
$j' \leftarrow \mathsf{A}(\mathsf{st}, (t_j, \sigma_j), (t_{\phi(i)}, \sigma_{\phi(i)})_{i\in\mathcal{Q}})$
$j' \leftarrow \mathsf{A}(\mathsf{st}, (t_j, \sigma_j), (t_k, \sigma_k), (t_{\phi(i)}, \sigma_{\phi(i)})_{i\in\mathcal{Q}})$
return $q_0 - q_1 \geq m$ and $j' = j$
```

```
Oracle $\mathrm{USER}_0$ in Hyb1, Hyb8, Hyb9

$q_0 := q_0 + 1$
$t_{q_0} \leftarrow_\$ \{0,1\}^\lambda$
if $\mathsf{H}_t(t_{q_0})$ was queried then abort
$(T'_{q_0}, \mathsf{st}_{q_0}) \leftarrow \mathsf{OSPP}.\mathsf{User}_0(X, t_{q_0})$
$r_{q_0} \leftarrow_\$ \mathbb{Z}_p^*$
$T_{q_0} := \mathsf{H}_t(t_{q_0});\ \ T'_{q_0} := r_{q_0}^{-1} \cdot T_{q_0}$
$T'_{q_0} \leftarrow_\$ \mathbb{G};\ \ T_{q_0} := \mathsf{H}_t(t_{q_0}) := r_{q_0} T'_{q_0}$
$\mathsf{st}_{q_0} := (\mathsf{pp}, r_{q_0}, t_{q_0}, T'_{q_0})$
$\mathcal{Q} := \mathcal{Q} \cup \{q_0\}$
return $(q_0, T'_{q_0})$
```

```
Oracle $\mathrm{USER}_1(i, (s_i, W'_i, \pi_i))$ in Hyb0, Hyb4, Hyb6, Hyb7

if $i \notin \mathcal{Q}$ then return $\perp$
// Below: $\sigma_i \leftarrow \mathsf{OSPP}.\mathsf{User}_1(\mathsf{st}_i, (s_i, W'_i, \pi_i))$
$(X, r_i, t_i, T'_i) := \mathsf{st}_i$
$S'_i := \mathsf{H}_s(T'_i, s_i)$
if not $\Pi_{\mathsf{DLEQ2}}.\mathsf{Verify}(\mathsf{pcrs}, (X, T'_i, S'_i, W'_i), \pi_i)$ then
    return $\perp$
$S_i := r_i S'_i;$   $S_i \leftarrow_\$ \mathbb{G}$
if not $\mathsf{R}_{\mathsf{DLEQ2}}((x, y), (X, T'_i, S'_i, W'_i))$ then abort
  $(x', y') \leftarrow \mathsf{Ext}_i(\mathsf{ptd}, (X, T'_i, S'_i, W'_i), \pi)$
$W := rW';$   $W := x\mathsf{H}_t(t) + yS$
$\sigma_i := (S_i, W_i)$
if $\sigma_i \neq \perp$ then
  $\mathcal{Q} := \mathcal{Q} \setminus \{i\};\ \ q_1 := q_1 + 1$
return $\sigma_i$
```

Fig. 17. Summary of hybrid changes for proof of unlinkability of OSPP in Theorem 7. We recall that $\mathcal{S}_X$ denotes the symmetric group of X, i.e., the group of all permutations of X.

Hyb_1 We unroll $\mathsf{OSPP}.\mathsf{User}_0$ inside the oracle USER_0. This hybrid is perfectly indistinguishable from Hyb_0.

Hyb_2 In this hybrid, we return 0 if the set $\mathcal{Q}$ is empty after having removed j (that is, if $\mathcal{Q} = \{j\}$). This hybrid is perfectly indistinguishable from Hyb_1, since if $|\mathcal{Q}| = 1 = q_0 - 1 - q_1$, the winning condition $q_0 - q_1 \geq m$ cannot be satisfied.

Hyb_3 We now sample another element k from $\mathcal{Q}$, and shuffle the others. The distribution is the same as in the previous hybrid.

Hyb_4 We now use the knowledge extractor, using a sequence of hybrids, to recover the witness. This hybrid is indistinguishable from the previous by the knowledge soundness of the proof system. Hence, we have:

$$q_0 \cdot \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}},\mathsf{B},\mathsf{Ext}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{OSPP},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_4}_{\mathsf{OSPP},\mathsf{A}}(\lambda) \right|,$$

where q_0 is the total number of calls to $\mathsf{OSPP}.\mathsf{User}_1$.

Hyb_5 This hybrid aborts if the extracted witnesses at positions j and k differ, and then sets $W_j \coloneqq x_k \mathsf{H}_t(t_j) + y_k S_j$ and $W_k \coloneqq x_j \mathsf{H}_t(t_k) + y_j S_k$ (which does not change the distributions).
If the game aborts at this step, by soundness of the proof system we would have that $X = x_j G + y_j H = x_k G + y_k H$ 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 a PPT adversary B for $\mathrm{DLOG}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$: the adversary B obtains a group description $\Gamma \coloneqq (\mathbb{G}, p, \tilde{G})$ and a challenge $\tilde{H}$. It sets $G \coloneqq \tilde{G}$ and $H \coloneqq \tilde{H}$ and runs exactly as per Hyb_4. 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_5 aborts.

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 left 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 a PPT adversary A for which the outcome of the two hybrids is different, then it is possible to construct a distinguisher B for $\mathrm{DDH}^{\beta}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ by exploiting the random self-reducibility property of DDH. The adversary B takes as input the group description together with a tuple $(P, A \coloneqq aP, B \coloneqq bP, C) \in \mathbb{G}^4$ where $a, b \leftarrow_\$ \mathbb{Z}_p$ and has to distinguish $C \coloneqq abP$ (the case $\beta = 0$) from a uniformly distributed element over $\mathbb{G}$ (the case $\beta = 1$). B runs the game as per Hyb_6, except in $\mathsf{OSPP.User}_0$, instead of sampling T'_{q_0} uniformly at random from $\mathbb{G}$, B sets them as $T'_{q_0} \coloneqq \gamma_{q_0} P$ where $\gamma_{q_0} \leftarrow_\$ \mathbb{Z}_p$. Instead of computing $T_{q_0} \coloneqq r_{q_0} T'_{q_0}$, B sets $T_{q_0} \coloneqq \gamma_{q_0}\alpha_{q_0} A + \gamma_{q_0}\alpha'_{q_0} P = (\alpha_{q_0} a + \alpha'_{q_0})\gamma_{q_0} P = (\alpha_{q_0} a + \alpha'_{q_0}) T'_{q_0}$, i.e. implicitly setting $r_{q_0} \coloneqq \alpha_{q_0} a + \alpha'_{q_0}$. For every query $\mathsf{H}_s(T'_{q_0}, s)$, the adversary B samples $\beta_{q_0,s}, \beta'_{q_0,s}$ uniformly at random and programs $\mathsf{H}_s(T'_{q_0}, s) \coloneqq \gamma_{q_0}\beta_{q_0,s} B + \gamma_{q_0}\beta'_{q_0,s} P = (\beta_{q_0,s} b + \beta'_{q_0,s})\gamma_{q_0} P$. Finally, when computing S at index i, B sets $S_i \coloneqq \alpha_i \beta_{i,s_i}\gamma_i C + \alpha_i \beta'_{i,s_i}\gamma_i A + \alpha'_i \beta_{i,s_i}\gamma_i B + \alpha'_i\beta'_{i,s_i}\gamma_i P$. If C is uniformly random then S_i is uniformly distributed as well, and therefore its distribution didn't change from 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. Therefore, it follows that the advantage in distinguishing the two hybrids is:

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

for any PPT adversary A.

Hyb_8 This hybrid aborts if the adversary has already queried $\mathsf{H}_t(t_{q_0})$ before running $\mathsf{OSPP.User}_0$. This happens with negligible probability:

$$\frac{q_0}{2^\lambda} \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_8}_{\mathsf{OSPP},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_7}_{\mathsf{OSPP},\mathsf{A}}(\lambda) \right|.$$

Hyb_9 In this hybrid, we program the random oracle so that T'_{q_0} is sampled uniformly at random, and we program the random oracle so that $T_{q_0} = \mathsf{H}_t(t_{q_0}) \coloneqq r_{q_0} T'_{q_0}$. The distribution is unchanged.

Hyb_{10} 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. This hybrid is perfectly indistinguishable from Hyb_5.

Now, it is sufficient to bound the probability of success adversary in Hyb_{10}. Since the index that the adversary needs to guess j is independent of the challenge it receives, we conclude that $\mathsf{Adv}^{\mathsf{Hyb}_{10}}_{\mathsf{OSPP},\mathsf{A}}(\lambda) \leq 1/(q_0 - q_1)$. Therefore:

$$\Pr\left[\mathrm{UNLINK}_{\mathsf{OSPP},\mathsf{A},m}(\lambda) = 1\right] \leq \frac{1}{m} + q_0 \cdot \mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{dlog}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) + \frac{q_0}{2^\lambda}$$

which completes the proof. $\square$

G Security proofs for PMBT

G.1 Unforgeability

We prove the following theorem using a hybrid argument. The key difference to OSPP (cf. Appendix F.1) 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 8. *If CTGDH holds for* GrGen *and* Π_{DLEQ2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathit{DLEQ2}}$, *then* $\mathsf{PMBT}[\mathsf{GrGen},\Pi_{\mathsf{DLEQ2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PMBT},\ell}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathit{DLEQ}}}(\lambda).$$

Proof. We prove this theorem using a hybrid argument, similar to the proof of Theorem 6. First, we replace the proving algorithm $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$ by the zero-knowledge simulator $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}$ (which defines an hybrid Hyb_1), 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 a PPT adversary A that has non-negligible advantage in the hybrid with simulated proofs, then we can construct a PPT adversary B that has non-negligible advantage in the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$.

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 $X_b := A + y_b H$ and $X_{1-b} := x_{1-b} G + y_{1-b} H$. With overwhelming probability it holds that $X_0 \neq X_1$. Then, it sets $\mathbf{X} := (X_0, X_1)$ and it runs $\mathsf{A}(\mathbf{X})$. Note that $\mathbf{X}$ is distributed as in the previous hybrid Hyb_1. The adversary B responds to the oracles H_t, SIGN, and READ using the oracles TARGET, HELP, and DDH (cf. Figure 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 to the signing oracles $\textsc{Sign}'(\hat{b}, T')$ and $\textsc{Sign}(T')$, the adversary B proceeds as follows:
  - it samples $s \leftarrow\!\!\$\ \{0,1\}^\lambda$ and defines $S' := \mathsf{H}_s(T', s)$;
  - if $\hat{b} = 1 - b$, it computes $W' = x_{1-b} T' + y_{1-b} S'$; else (i.e., $\hat{b} = b$), it invokes the oracle $Z := \textsc{Help}(T')$ and sets $W' = Z + y_b S'$;

  - finally, returns (s, W', π), where the proof π is simulated.

– to any query of the form $\textsc{Read}(t, (S, W))$, B (who knows the secrets y_b, x_{1-b}, and y_{1-b}) lets $X := W - y_b S$, and defines $bool_b := \textsc{Ddh}(t, X)$ and $bool_{1-b} := (W = x_{1-b}\mathsf{H}_t(t) + y_{1-b}S)$. It returns $\perp$ 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, $\textsc{Sign}$, $\textsc{Sign}'$, and $\textsc{Read}$ are identical to the ones of Hyb_1. At the end of the execution, A returns $\ell + 1$ tuples $(t_i, (S_i, W_i)) \in \{0,1\}^\lambda \times \mathbb{G}^2$. If the $\ell+1$ forgeries are valid w.r.t. the bit b (that is, B correctly guessed the bit of the forgery at the beginning), it returns $(t_i, W_i - y_b S_i)_{i \in [\ell+1]}$. Otherwise, it returns $\perp$. 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 (sampled unif. at random from $\{0,1\}$). This shows that:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PMBT},\ell}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQ2}},\mathsf{R}_{\mathsf{DLEQ2}}}(\lambda),$$

and this concludes the proof.

G.2 Unlinkability

We prove that PMBT is **2**-unlinkable instead of 1-unlinkable (cf. Definition 2), 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, at most halving its search space.

Theorem 9. *If DDH holds for* GrGen *and* Π_{DLEQOR2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLEQOR2}}$*, then* $\mathsf{PMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQOR2}}]$ *is 2-unlinkable.*

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 7. Since the user algorithms are the same (besides the DLOEQOR instead of DLEQ proof) as in Construction 2, we explicit the key difference in the reduction below, and refer to Fig. 17 and the proof of Theorem 7 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 7, 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

$$\mathfrak{p} = \sum_{\substack{b'=0,1 \\ U_{b'} \neq \emptyset}} \Pr[b' = b] \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda) = \sum_{\substack{b'=0,1 \\ U_{b'} \neq \emptyset}} \frac{|U_{b'}|}{|U|} \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda)$$
$$= \sum_{\substack{b'=0,1 \\ U_{b'} \neq \emptyset}} \frac{|U_{b'}|}{k_0 - k_1} \cdot \frac{1}{|U_{b'}|} + \mathsf{negl}(\lambda) \leq \frac{2}{k_0 - k_1} + \mathsf{negl}(\lambda) \leq \frac{2}{m} + \mathsf{negl}(\lambda) .$$

G.3 Privacy of the metadata bit

Theorem 10. *If DDH holds for the group generator* GrGen*, and* Π_{DLEQOR2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLEQOR2}}$ *then* $\mathsf{PMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQOR2}}]$ *provides private metadata bit*

```
Oracle SIGN'($\hat b, T'$) in Hyb_0, Hyb_1, Hyb_2
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat b}T' + y_{\hat b}S'$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}(\mathsf{pcrs}, (X_0, X_1, T', S', W'), (x_{\hat b}, y_{\hat b}))$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}(\mathsf{ptd}, (X_0, X_1, T', S', W'))$
return $(s, W', \pi)$
```

```
Oracle SIGN($T'$) in Hyb_1($\lambda$), Hyb_2($\lambda$), Hyb_3($\lambda$)
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}(\mathsf{pcrs}, (X_0, X_1, T', S', W'), (x_b, y_b))$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}(\mathsf{ptd}, (X_0, X_1, T', S', W'))$
return $(s, W', \pi)$
```

```
Oracle SIGN'($\hat b, T'$) in Hyb_4($\lambda$), Hyb_5($\lambda$), Hyb_6($\lambda$)
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_{\hat b}T' + y_{\hat b}S'$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}(\mathsf{ptd}, (X_0, X_1, T', S', W'))$
return $(s, W', \pi)$
```

```
Oracle SIGN($T'$) in Hyb_4($\lambda$), Hyb_5($\lambda$), $\boxed{\mathsf{Hyb}_6(\lambda)}$
$s \leftarrow\$ \{0,1\}^\lambda$
if $\mathsf{H}_s(T', s)$ was queried then abort
$S' := \mathsf{H}(T', s)$
$W' := x_b T' + y_b S'$
$y' \leftarrow\$ \mathbb{Z}_p^*$;  $W' := x_b T' + y' S'$
$W' := x_{1-b} T' + y' S'$
$\boxed{W' := x_{1-b} T' + y_{1-b} S'}$
$\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}(\mathsf{ptd}, (X_0, X_1, T', S', W'))$
return $(s, W', \pi)$
```

Fig. 18. Hybrid changes for privacy of the metadata bit of PMBT in Theorem 10.

with advantage:

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBT}}(\lambda) \le \frac{O(q^2)}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 2\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQOR2}},\mathsf{R}_{\mathsf{DLEQOR2}}}(\lambda),$$

where q is the number of queries the adversary makes to H_s or Sign.

Proof. We consider the sequence of hybrids presented in Fig. 18 that transitions from $\mathrm{PMB}^b_{\mathsf{PMBT},\mathsf{A}}(\lambda)$ to $\mathrm{PMB}^{1-b}_{\mathsf{PMBT},\mathsf{A}}(\lambda)$ (see Definition 3). We argue that each pair of consecutive hybrids are indistinguishable, for any PPT adversary A. We do not explicitly write the verification oracle in the games since it will always return **true**.

Hyb_0 This is the game $\mathrm{PMB}^0_{\mathsf{AT},\mathsf{A}}(\lambda)$. Here, the adversary is provided the public parameters $\mathbf{X} := (X_0, X_1)$. 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, A outputs a bit b'; if $b = b'$, the adversary won.

Hyb_1 This hybrid replaces the way zero-knowledge proofs are generated in Sign and Sign′: instead of using the proving algorithm $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$, we use the zero-knowledge simulator $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}$.
If there exists a PPT adversary A whose output is different between the two games, then it is possible to construct a distinguisher for the zero-knowledge proof system Π_{DLEQOR2}: consider the PPT adversary B for the game $\mathrm{ZK}^\beta_{\Pi_{\mathsf{DLEQOR2}},\mathsf{R}_{\mathsf{DLEQOR2}},\mathsf{B}}(\lambda)$ (cf. Fig. 2) that, given as input the CRS pcrs, runs $(\mathbf{X}, (\mathbf{x}, \mathbf{y})) \leftarrow \mathsf{PMBT}.\mathsf{KeyGen}((\Gamma, \mathsf{pcrs}))$ and then invokes the adversary

$\mathsf{A}((\mathit{\Gamma}, \mathsf{pcrs}), \mathbf{X})$. All random oracles queries are performed exactly as per Hyb_2, except for signing queries. For each query to $\textsc{Sign}$ and $\textsc{Sign}'$, after generating the values s, S', and W' as per Hyb_0, the proof π is generated via the proving oracle: $\pi \leftarrow \textsc{Prove}((\mathbf{X}, T', S', W'), (\hat{b}, x_{\hat{b}}, y_{\hat{b}}))$, where $\hat{b} \in \{0,1\}$ is specified within the signing oracle ($\hat{b} = b$ in $\textsc{Sign}$). At the end of its execution, A returns a guess b'; B outputs b'.

If the $\textsc{Prove}$ oracle outputs proofs via $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$, the game is identical to Hyb_0. If $\textsc{Prove}$ outputs proofs via $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}$, the game is identical to Hyb_1. It follows that, for any PPT adversary A, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in Π_{DLEQOR2} with adversary B, that is:

$$\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLEQOR2}}, \mathsf{R}_{\mathsf{DLEQOR2}}, \mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PMBT},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PMBT},\mathsf{A}}(\lambda) \right| .$$

Hyb_2 If, during any of the signing queries, the oracle H_s has already 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 $s \in \{0,1\}^\lambda$ between the signing oracles, or via a direct query to H_s. For a PPT adversary A making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles H_s, $\textsc{Sign}$, or $\textsc{Sign}'$, the probability that the game aborts is at most $(q-1)(q-2)/2^\lambda$. In fact, for the second query (to $\textsc{Sign}$ or $\textsc{Sign}'$), we have chance $1/p$ of hitting the string s selected during the first query, and so on: the i-th query has chance $(i-1)/p$ of selecting a s that has been previously used. It follows that:

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

Hyb_3 We now change the way W' is computed: during the key generation, we sample an additional element $y' \leftarrow\$\, \mathbb{Z}_p$, and for any query to the random oracle $\textsc{Sign}$ we construct W' as $x_b T' + y' S'$ instead of $x_b T' + y_b S'$. The proof π gets simulated as before.

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

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

The adversary B proceeds as per Hyb_2, embedding the challenge in the public key and oracle replies. It fixes $H \coloneqq P$, and instead of generating $X_b \coloneqq x_b G + y_b H$, it constructs it as $X_b \coloneqq x_b G + A$. Then, it runs the adversary A. The adversary A will make queries to any of the random oracles H_t and H_s, that are answered as before. We replace queries to the signing oracles in the following way:

- for any query $\textsc{Sign}$, 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 $W' \coloneqq x_b T' + \beta_i C + \alpha_i A$ and produces the proof π using the simulator. B returns (s, W', π)

– for any query $\textsc{Sign}'$ with $\hat{b} = b$, 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 $W' \coloneqq x_b T' + \alpha_i A$, and simulates the proof. It returns (s, W', π).
– any query to $\textsc{Sign}'$ with $\hat{b} = 1 - b$ is handled exactly as per the previous hybrid.

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 $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Sim}$ fails), B will distinguish a random tuple from a DDH tuple. It follows therefore that:

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

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:

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

At this point we note that the oracle $\textsc{Sign}$ 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 $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$ (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 $\mathrm{PMB}^\beta_{\mathsf{PMBT},\mathsf{A}}(\lambda)$ is:

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

where q is the number of queries to the signing oracles or to the random oracle H_s.

H Security proofs for PPB

In this section, we argue one-more unforgeability and 2-unlinkability of Construction 4.

H.1 Unforgeability

Theorem 21. *If CTGDH holds for* GrGen*, and* Π_{DLOG} *is a zero-knowledge proof system for relation* Π_{DLOG}*, then* $\mathsf{PPB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOG}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PPB},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOG}},\mathsf{R}_{\mathsf{DLOG}}}(\lambda).$$

```
Game $\mathrm{UNLINK}_{\mathsf{PPB},\mathsf{A},m}(\lambda)$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_1$
$\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLOG}}.\mathsf{Setup}(\Gamma)$
$(\mathsf{st}, (X, \pi)) \leftarrow \mathsf{A}((\mathsf{pcrs}, \Gamma))$
$x \leftarrow \mathsf{Ext}(\mathsf{ptd}, \pi)$
if not $\mathsf{R}_{\mathsf{DLOG}}(X, x)$ then abort
$q_0 := 0; q_1 := 0; \mathcal{Q} := \emptyset$
$(\mathsf{st}, \{W'_i\}_{i \in \mathcal{Q}}) \leftarrow \mathsf{A}^{\mathrm{USER}_0(),\mathrm{USER}_1(\cdot,\cdot)}(\mathsf{st})$
if $\mathcal{Q} = \emptyset$ then return $0$
$j \leftarrow\!\$\, \mathcal{Q}$;  $\mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$\sigma_j \leftarrow \mathsf{AT.User}_1(\mathsf{st}_j, W'_j)$
$\forall i \in \mathcal{Q}, \sigma_i \leftarrow \mathsf{AT.User}_1(\mathsf{st}_i, W'_i)$
$\phi \leftarrow \mathcal{S}_{\mathcal{Q}}$
$j' \leftarrow \mathsf{A}(\mathsf{st}, (t_j, \sigma_j), (t_{\phi(i)}, \sigma_{\phi(i)})_{i \in \mathcal{Q}})$
return $q_1 - q_0 \geq m$ and $j' = j$
```

```
Oracle $\mathrm{USER}_0()$
$q_0 := q_0 + 1$
$t_{q_0} \leftarrow\!\$\, \{0,1\}^\lambda$
$(T'_{q_0}, \mathsf{st}_{q_0}) \leftarrow \mathsf{AT.User}_0(\mathsf{pp}, t_{q_0})$
$\mathcal{Q} := \mathcal{Q} \cup \{q_0\}$
return $(q_0, T'_{q_0})$
```

```
Oracle $\mathrm{USER}_1(j, W')$
if $j \notin \mathcal{Q}$ then return $\perp$
$\sigma_j \leftarrow \mathsf{AT.User}_1(\mathsf{st}_j, W')$
$\mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$q_1 := q_1 + 1$
return token
```

```
$\mathsf{AT.User}_0(\mathsf{pp} = (X, \pi), t))$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_3$
$r, \rho \leftarrow\!\$\, \mathbb{Z}_p^*$
$T := \mathsf{H}_t(t)$
$T' := r(T - \rho G)$
$T' \leftarrow\!\$\, \mathbb{G}$
$\mathsf{st} := (\mathsf{p}, r, \rho, t)$
return $(T', \mathsf{st})$
```

```
$\mathsf{AT.User}_1(\mathsf{pp}, r, \rho, t), W')$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_2$, $\mathsf{Hyb}_4$
$(X, \pi) := \mathsf{pp}$
$\sigma := r^{-1} W' + \rho X$
$P := W' - xT'$
$\sigma := x\mathsf{H}_t(t) + r^{-1} P$
if $P \neq 0$ then $\sigma \leftarrow\!\$\, \mathbb{G}$
return $\sigma$
```

Fig. 19. Hybrid changes for unlinkability of PPB in Theorem 22.

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, and the zero-knowledge proof sent within pp can be simulated.

H.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 22. *Let* GrGen *be a group generator. If* Π_{DLOG} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathsf{DLOG}}$*, then* $\mathsf{PPB}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ}}]$ *is 2-unlinkabile.*

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_0 The first hybrid is the unlinkability game.

Hyb_1 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.

$$\mathsf{Adv}^{\mathrm{ksnd}}_{\Pi_{\mathsf{DLOG}},\mathsf{R}_{\mathsf{DLOG}},\mathsf{B}}(\lambda) \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PPB},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PPB},\mathsf{A}}(\lambda) \right|.$$

Hyb_2 In this hybrid we compute the value σ in $\mathsf{AT.User}_1$ 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_3 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_4 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.

$$\begin{aligned}\Pr\left[\mathrm{UNLINK}_{\mathsf{PPB,A},m}(\lambda)\right] &= \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \Pr\left[\mathrm{UNLINK}_{\mathsf{PPB,A},m}(\lambda) \mid j \in U_d\right] \Pr[j \in U_d] \\ &\leq \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \frac{1}{|U_d|}\frac{|U_d|}{|U|} + \mathsf{negl}(\lambda) \leq \sum_{\substack{d=0,1\\U_d\neq\emptyset}} \frac{1}{q_0 - q_1} + \mathsf{negl}(\lambda) \\ &\leq \frac{2}{m} + \mathsf{negl}(\lambda).\end{aligned}$$

I Security proofs for PMBTB

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

I.1 Unforgeability

Theorem 13. *If CTGDH holds for the group generator algorithm* GrGen *and* Π_{DLOGAND2} *is a zero-knowledge proof system for* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *is one-more unforgeable with advantage:*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PMBTB},\ell}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda).$$

Proof. We prove this theorem using a hybrid argument. First, we replace the proving algorithm by the zero-knowledge simulator, and then show a reduction to the CTGDH problem. Let ℓ be an integer.

Hyb_0 This is the game $\mathrm{OMUF}_{\mathsf{PMBTB,A},\ell}(\lambda)$: the adversary is provided with the public parameters $(\mathbf{X}, \pi)$. The adversary has access to the signing oracle SIGN, the token validity oracle VERIFY, the metadata extraction oracle READ, and the random oracles H_s (used for the response of the issuer) and H_t (user for bliding the message by the user). At the end of its execution, it outputs $\ell + 1$ tokens for the same bit b'.

Hyb_1 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 a PPT adversary A whose advantage is different between the two games, then it is possible to construct a distinguisher B for the underlying zero-knowledge of the proof system. The adversary simply generates the proofs via the PROVE oracle in $\mathrm{ZK}^{\beta}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda)$ to generate the proofs: if $\beta = 0$ we're in Hyb_0; if $\beta = 1$ we're in Hyb_1. We have:

$$\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}},\mathsf{B}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PMBTB},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)\right|.$$

We will now prove that if there is an adversary A that has non-negligible advantage $\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)$, then we can construct an adversary B that has non-negligible advantage in winning 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. First, B 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 $X_b := A + y_b H$, and $X_{1-b} := x_{1-b}G + y_{1-b}H$, and simulates the proof π. The public parameters are $(\mathbf{X}, \pi) = ((X_0, X_1), \pi)$. It runs $\mathsf{A}(\mathbf{X}, \pi)$. Note that $\mathbf{X}$ is distributed as in Hyb_1. We need now to specify how B answers oracle queries. The adversary B overrides the queries to the oracles 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 $\textsc{Sign}(b', (T'_0, T'_1))$, it proceeds as follows.
  - If $b' = 1-b$, it follows the issuance protocol described in Figure 10, 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 $\textsc{Read}(t, (S_0, S_1, W_0, W_1))$, 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 $\textsc{Read}$ are identical to the ones of Hyb_1. At the end of the execution, A returns $\ell + 1$ tuples $(t_i, (S_i, W_i)) \in \{0,1\}^\lambda \times \mathbb{G}^2$. If the $\ell + 1$ forgeries are valid w.r.t the bit b (that is, B correctly guessed the bit of the forgery at the beginning), it returns $(t_i, W_i - y_b S_i)_{i \in [\ell+1]}$. Otherwise, it returns $\perp$. We claim that the adversary B wins the game $\mathrm{CTGDH}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ every time B guesses correctly the bit b (sampled uniformly at random) of the forgery of A. By the winning condition of game Hyb_1, A wins 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). By winning condition of unforgeability, A only called the oracle $\textsc{Read}$ at most ℓ times; therefore $\textsc{Help}$ was called at most ℓ times. Finally, this shows that:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{PMBTB},\mathsf{A},\ell}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ctgdh}}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda) + \mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}},\mathsf{A}}(\lambda).$$

and this concludes the proof.

I.2 Unlinkability

Theorem 14. *If DDH holds for the group generator* GrGen *and* Π_{DLOGAND2} *is a knowledge-sound proof system for relation* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *is 3-unlinkabile.*

```
Game $\mathrm{UNLINK}_{\mathsf{PMBTB},\mathsf{A},m}(\lambda)$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_1$
$\mathit{\Gamma} := (\mathbb{G}, p, G, H) \leftarrow \mathsf{GrGen}(1^\lambda)$
$(\mathsf{st}, (\mathbf{X}, \pi)) \leftarrow \mathsf{A}(\mathit{\Gamma})$
$(x_0, y_0) \leftarrow \mathsf{Ext}(\pi)$
if not $R((x_0, y_0), X_0)$ then abort
$q_0 := 0; q_1 := 0; \mathcal{Q} := \emptyset$
$(\mathsf{st}, \{W'_i\}_{i \in \mathcal{Q}}) \leftarrow \mathsf{A}^{\textsc{User}_0(), \textsc{User}_1(\cdot,\cdot)}(\mathsf{st})$
if $\mathcal{Q} = \emptyset$ then return $0$
$j \leftarrow\!\$\, \mathcal{Q}$;  $\mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$\mathsf{token}_j := \mathsf{PMBTB.User}_1(\mathsf{st}_j, W'_j)$
$\forall i \in \mathcal{Q}, \mathsf{token}_i := \mathsf{PMBTB.User}_1(\mathsf{st}_i, W'_i)$
$\phi \leftarrow \mathcal{S}_{\mathcal{Q}}$
$j' \leftarrow \mathsf{A}(\mathsf{st}, \mathsf{token}_j, \{\mathsf{token}_{\phi(i)}\}_{i \in \mathcal{Q}})$
return $q_0 - q_1 \geq m$ and $j' = j$
```

```
$\mathsf{PMBTB.User}_1((\mathsf{pp}, \{r_d, \rho_d, T'_d\}_{d=0,1}, t), (s, W'))$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_2$, $\mathsf{Hyb}_3$, $\boxed{\mathsf{Hyb}_4}$
$(X_0, X_1, \pi_0, \pi_1) := \mathsf{pp}$
$S_0 = r_0^{-1}\mathsf{H}_s(T'_0, s) + \rho_0 H$;  $S_0 \leftarrow\!\$\, \mathbb{G}$;  $W_0 := r_0^{-1}W' + \rho_0 X_0$
$S_1 = r_1^{-1}\mathsf{H}_s(T'_1, s) + \rho_1 H$  $\boxed{S_1 \leftarrow\!\$\, \mathbb{G};}$  $W_1 := r_1^{-1}W' + \rho_1 X_1$
$P_0 := W' - x_0 T'_0 - y_0 \mathsf{H}_s(T'_0, s)$;  $W_0 := x_0 \mathsf{H}_t(t) + y_0 S_0 + r_0^{-1} P_0$
$P_1 := W' - x_1 T'_1 - y_1 \mathsf{H}_s(T'_1, s)$;  $W_1 := x_1 \mathsf{H}_t(t) + y_1 S_1 + r_1^{-1} P_1$
$\sigma := (S_0, S_1, W_0, W_1)$
return $\sigma$
```

```
$\mathsf{PMBTB.User}_1((\mathsf{pp}, \{r_d, \rho_d, T'_d\}_{d=0,1}, t), (s, W'))$ in $\mathsf{Hyb}_7$, $\mathsf{Hyb}_8$
$(X_0, X_1, \pi_0, \pi_1) := \mathsf{pp}$
$S_0 \leftarrow\!\$\, \mathbb{G}$
$S_1 \leftarrow\!\$\, \mathbb{G}$
$P_0 := W' - x_0 T'_0 - y_0 \mathsf{H}_s(T'_0, s)$;  $W_0 := x_0 \mathsf{H}_t(t) + y_0 S_0 + r_0^{-1} P_0$
$P_1 := W' - x_1 T'_1 - y_1 \mathsf{H}_s(T'_1, s)$;  $W_1 := x_1 \mathsf{H}_t(t) + y_1 S_1 + r_1^{-1} P_1$
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 $\sigma$
```

```
Oracle $\textsc{User}_0()$
$q_0 := q_0 + 1$
$t_{q_0} \leftarrow\!\$\, \{0,1\}^\lambda$
$(T'_{q_0}, \mathsf{st}_{q_0}) \leftarrow \mathsf{PMBTB.User}_0(\mathsf{pp}, t_{q_0})$
$\mathcal{Q} := \mathcal{Q} \cup \{q_0\}$
return $(q_0, T'_{q_0})$
```

```
$\mathsf{PMBTB.User}_0((X, \pi), t))$ in $\mathsf{Hyb}_0$, $\mathsf{Hyb}_5$, $\mathsf{Hyb}_6$
$T := \mathsf{H}_t(t)$
$r_0, \rho_0 \leftarrow\!\$\, \mathbb{Z}_p^*$
$r_1, \rho_1 \leftarrow\!\$\, \mathbb{Z}_p^*$
$T'_0 := r_0(T - \rho_0 G)$
$T'_1 := r_1(T - \rho_1 G)$
$T'_0 \leftarrow\!\$\, \mathbb{G}$
$T'_1 \leftarrow\!\$\, \mathbb{G}$
$\mathsf{st} := (\mathsf{pp}, \{r_d, \rho_d\}_{d=0,1}, t)$
return $((T'_0, T'_1), \mathsf{st})$
```

```
Oracle $\textsc{User}_1(j, W')$
if $j \notin \mathcal{Q}$ then return $\bot$
$\mathsf{token} \leftarrow \mathsf{PMBTB.User}_1(\mathsf{st}_j, W')$
$\mathcal{Q} := \mathcal{Q} \setminus \{j\}$
$q_1 := q_1 + 1$
return $\mathsf{token}$
```

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

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 uniformly random, and at most one of the values $W_0, W_1 \in \mathbb{G}$ 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.

Hyb_0 This is the execution where the adversary interacts with an honest user side for the token issuance.

Hyb_1 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.

$$\mathsf{Adv}^{\text{ksnd}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PMBTB},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)\right|.$$

Hyb_2 In this hybrid, we compute W_0 and W_1 in a different but equivalent way in $\mathsf{PMBTB.User}_1$. 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_3 In this hybrid, we sample S_0 uniformly at random. We show that if there is an adversary B that distinguishes Hyb_2 and Hyb_3, 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 $\mathsf{PMBTB.User}_1$, 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_2, and if cP is a random element, then the execution is Hyb_3. Hence,

$$\mathsf{Adv}^{\text{ddh}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\mathsf{PMBTB},\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_3}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)\right|.$$

Hyb_4 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_3 and Hyb_4, then we can construct an adversary that breaks DDH.

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

Hyb_5 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_4 and Hyb_5 have exactly the same distribution.

Hyb_6 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_7 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_6 and Hyb_7 are the same.

Hyb_8 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_8 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,

$$\begin{aligned}\Pr\left[\mathrm{UNLINK}_{\mathsf{PMBTB,A},m}(\lambda)\right] &= \sum_{\substack{d=1,2,3\\U_d\neq\emptyset}} \Pr\left[\mathrm{UNLINK}_{\mathsf{PMBTB,A},m}(\lambda) \mid j \in U_d\right] \Pr[j \in U_d] \\ &\leq \sum_{\substack{d=1,2,3\\U_d\neq\emptyset}} \frac{1}{|U_d|}\frac{|U_d|}{|U|} + \mathsf{negl}(\lambda) \leq \sum_{\substack{d=1,2,3\\U_d\neq\emptyset}} \frac{1}{q_0 - q_1} + \mathsf{negl}(\lambda) \\ &\leq \frac{3}{m} + \mathsf{negl}(\lambda).\end{aligned}$$

I.3 Privacy of the metadata bit

Theorem 15. *If DDH holds for the group generator* GrGen *and* Π_{DLOGAND2} *is a zero-knowledge proof system for relation* $\mathsf{R}_{\mathsf{DLOGAND2}}$*, then* $\mathsf{PMBTB}[\mathsf{GrGen}, \Pi_{\mathsf{DLOGAND2}}]$ *provides private metadata bit with advantage:*

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBTB}}(\lambda) \leq \frac{O(q^2)}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 4\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda),$$

where q is the number of queries the adversary makes either to H_s or Sign*.*

Proof. We consider a sequence of hybrids (summarized in Fig. 21) that transitions from an execution of $\mathrm{PMB}^{\beta}_{\mathsf{PMBTB,A}}(\lambda)$ to an execution of $\mathrm{PMB}^{1-\beta}_{\mathsf{PMBTB,A}}(\lambda)$ (see Definition 3). We argue that each pair of consecutive hybrids are indistinguishable for the adversary and thus that the advantage $\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBTB,A}}(\lambda)$ is negligible. We do not explicitly write the verification validity oracle in the games since in this construction this functionality is dummy.

Hyb_0 This is the game $\mathrm{PMB}^{0}_{\mathsf{PMBTB,A}}(\lambda)$: 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 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: instead of using the proving algorithm $\Pi_{\mathsf{DLOGAND2}}.\mathsf{Prove}$, we use the zero-knowledge simulator $\Pi_{\mathsf{DLOGAND2}}.\mathsf{Sim}$.
If there exists a PPT adversary A 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 PPT adversary B for the game $\mathrm{ZK}^{\beta}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}},\mathsf{B}}(\lambda)$ that generates X_0, X_1 as per $\mathsf{PMBTB.KeyGen}(1^\lambda)$ and uses the Prove oracle for the statement (X_0, X_1). At the end of its execution, A (and so B) return a guess b'.
If the Prove oracle outputs proofs via $\Pi_{\mathsf{DLOGAND2}}.\mathsf{Prove}$, the game is identical to Hyb_0, else the game is identical to Hyb_1. It follows that, for any PPT adversary A, the advantage in distinguishing the two hybrids is at most the advantage of zero-knowledge in Π_{DLOGAND2}, i.e.:

$$\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}},\mathsf{A}}(\lambda) \geq \left|\mathsf{Adv}^{\mathsf{Hyb}_1}_{\mathsf{PMBTB,A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_0}_{\mathsf{PMBTB,A}}(\lambda).\right|$$

```
Oracle $\textsc{Sign}'(\hat{b}, T')$ 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
$S'_{\hat{b}} \coloneqq \mathsf{H}(T'_{\hat{b}}, s)$
$W' \coloneqq x_{\hat{b}} T'_{\hat{b}} + y_{\hat{b}} S'_{\hat{b}}$
return $(s, W')$
```

```
Oracle $\textsc{Sign}(T')$ 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
$S'_b \coloneqq \mathsf{H}(T'_b, s)$
$W' \coloneqq x_b T'_b + y_b S'_b$
return $(s, W')$
```

```
Oracle $\textsc{Sign}'(\hat{b}, T')$ in $\mathsf{Hyb}_{4\text{–}6}(\lambda)$
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $\mathsf{H}_s(T'_{\hat{b}}, s)$ was queried then abort
$S'_{\hat{b}} \coloneqq \mathsf{H}(T'_{\hat{b}}, s)$
$W' \coloneqq x_{\hat{b}} T'_{\hat{b}} + y_{\hat{b}} S'_{\hat{b}}$
return $(s, W')$
```

```
Oracle $\textsc{Sign}(T')$ 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
$S'_b \coloneqq \mathsf{H}(T'_b, s)$
$W' \coloneqq x_b T'_b + y_b S'_b$
$y' \leftarrow\!\!\$\ \mathbb{Z}_p^*;\quad W' \coloneqq x_b T'_b + y' S'_b$
$W' \coloneqq x_{1-b} T'_{1-b} + y' S'_{1-b}$
$\boxed{W' \coloneqq x_{1-b} T'_{1-b} + y_{1-b} S'_{1-b}}$
return $(s, W')$
```

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

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'_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 a PPT adversary A making at most $q = \mathsf{poly}(\lambda)$ queries to any of the oracles H_s, $\textsc{Sign}$, or $\textsc{Sign}'$, the probability that the game aborts is at most $q(q-1)/2p$. It follows that:

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

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 $\mathsf{PMBTB.Sign}_0(\mathsf{pp}, \mathsf{sk}, b, \cdot)$ 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 $\mathrm{DDH}^{\beta}_{\mathsf{B,GrGen}}(\lambda)$. The adversary B wins every time the output of the two hybrids is different. The adversary B receives as input a DDH tuple $(P, A \coloneqq aP, B \coloneqq bP, C) \in \mathbb{G}^4$ such that $C = abP$ in the case $\mathrm{DDH}^0_{\mathsf{B,GrGen}}(\lambda)$ and $C \leftarrow\!\!\$\ \mathbb{G}$ in the case $\mathrm{DDH}^1_{\mathsf{B,GrGen}}(\lambda)$. 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_2, embedding the challenge in the public key and oracle replies. It fixes $H \coloneqq P$, and instead of generating $X_b \coloneqq x_b G + y_b H$, it constructs it as $X_b \coloneqq x_b G + A$. Then, it runs the adversary A. The adversary A will make queries to any of the random oracles H_t 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 to $\textsc{Sign}(T')$, 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 $\mathsf{H}_s(T'_b, s) = S'_b$ 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 to $\textsc{Sign}'(\hat{b}, T')$ with $\hat{b} = b$, 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').
- for any query to $\textsc{Sign}'(\hat{b}, T')$ with $\hat{b} = 1 - b$ 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. 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:

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

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

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

At this point we note that the oracle $\textsc{Sign}$ 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 $\mathsf{Hyb}_1 \rightarrow \mathsf{Hyb}_2$), and swap back the zero-knowledge simulator with the prover's algorithm $\Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}$ (via the same argument used for the transition $\mathsf{Hyb}_0 \rightarrow \mathsf{Hyb}_1$). Therefore, the advantage of an adversary A in winning the game $\mathrm{PMB}^{\beta}_{\mathsf{PMBTB},\mathsf{A}}(\lambda)$

$$\mathsf{Adv}^{\mathrm{pmb}}_{\mathsf{PMBTB},\mathsf{A}}(\lambda) \leq \frac{q(q-1)}{2^\lambda} + \frac{2}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen}}(\lambda) + 4\mathsf{Adv}^{\mathrm{zk}}_{\Pi_{\mathsf{DLOGAND2}},\mathsf{R}_{\mathsf{DLOGAND2}}}(\lambda)$$

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

J 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.

Construction 6. Let GrGen be a group generator algorithm; let Π_{DLEQ2} be a proof system for relation $\mathsf{R}_{\mathsf{DLEQ2}}$; let Π_{DLEQOR2} be a proof system for relation $\mathsf{R}_{\mathsf{DLEQOR2}}$; let $\mathsf{H}_t, \mathsf{H}_s$ be two random oracles $\{0,1\}^* \to \mathbb{G}$. We construct an anonymous token scheme CMBT defined by the following algorithms:

– $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{CMBT.Setup}(1^\lambda)$: invoke the group generator $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and the CRS generation algorithm of the underlying proof system $(\mathsf{pcrs}, \mathsf{ptd}) \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Setup}(\Gamma)$;

– $(\mathsf{pp}, \mathsf{sk}) \leftarrow \mathsf{CMBT.KeyGen}(1^\lambda)$: sample $x_0, x_1, \tilde{x}, y_0, y_1, \tilde{y}$ uniformly at random from $\mathbb{Z}_p$. Define:

$$\mathbf{X} := \begin{bmatrix} X_0 \\ X_1 \end{bmatrix} := \begin{bmatrix} x_0 G + y_0 H \\ x_1 G + y_1 H \end{bmatrix}, \qquad \tilde{X} := \tilde{x} G + \tilde{y} H.$$

Restart if $X_0 = X_1$. Set $\mathsf{sk} := ((x_0, y_0), (x_1, y_1), (\tilde{x}, \tilde{y}))$, and $\mathsf{pp} := (X_0, X_1, \tilde{X})$.

– $\sigma \leftarrow \langle \mathsf{CMBT.User}(\mathsf{pp}, t), \mathsf{CMBT.Sign}(\mathsf{sk}, b) \rangle$: illustrated in Figure 22.

– $\mathit{bool} \leftarrow \mathsf{CMBT.Verify}(\mathsf{sk}, t, \sigma)$: read σ as $(S, W, \tilde{W}) \in \mathbb{G}^3$. Return **true** if $\tilde{W} = \tilde{x}\mathsf{H}_t(t) + \tilde{y}S$; else, return **false**.

– $\mathsf{ind} \leftarrow \mathsf{CMBT.ReadBit}(\mathsf{sk}, t, \sigma)$: read σ as $(S, W, \tilde{W})$. Then:

(a) 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

(b) 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

(c) Else, return $\perp$.

J.1 Unforgeability

Theorem 23. *If CTGDH is hard for* GrGen*,* Π_{DLEQ2} *is a knowledge-sound proof system for relation* Π_{DLEQ2}*, and* Π_{DLEQOR2} *is a knowledge-sound proof system for relation* Π_{DLEQOR2}*, then* $\mathsf{CMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}, \Pi_{\mathsf{DLEQOR2}}]$ *is one-more unforgeable.*

One-more unforgeability follows from the one-more unforgeability of Construction 3. We can construct an adversary for Constr. 3 by interacting with an adversary for Constr. 6 in the same way as the reduction in the proof of Theorem 8 since the user messages in both constructions are the same, and the issuer's response in Constr. 3 is a subset of the issuer's response in Constr. 6.

J.2 Unlinkability

Theorem 24. *If DDH is hard for* GrGen*,* Π_{DLEQ2} *is a knowledge-sound proof system for relation* Π_{DLEQ2}*, and* Π_{DLEQOR2} *is a knowledge-sound proof system for relation* Π_{DLEQOR2}*, then* $\mathsf{CMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}, \Pi_{\mathsf{DLEQOR2}}]$ *is 2-unlinkable.*

Unlinkability of this construction follows from the unlinkability of Constr. 3. We can construct an adversary for Constr. 3 by interacting with an adversary for Constr. 6 in the same way as the reduction in the proof of Theorem 9 since the user messages in both constructions are the same, and the issuer's response in Constr. 3 is a subset of the issuer's response in Constr. 6.

J.3 Privacy of metadata bit

Theorem 25. *If DDH holds for the group generator* GrGen Π_{DLEQ2} *is a knowledge-sound proof system for relation* Π_{DLEQ2}*, and* Π_{DLEQOR2} *is a knowledge-sound proof system for relation* Π_{DLEQOR2}*, then* $\mathsf{CMBT}[\mathsf{GrGen}, \Pi_{\mathsf{DLEQ2}}, \Pi_{\mathsf{DLEQOR2}}]$ *provides private metadata bit.*

Private metadata bit here follows closely the proof of Theorem 10. The only difference is that we need to handle verification queries from the adversary. Since 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.

```
$\mathsf{CMBT.User}(\mathsf{pp}, t)$                                      $\mathsf{CMBT.Sign}(\mathsf{pp}, \mathsf{sk}, b)$

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

                        $\xrightarrow{T'}$

                                      $\mathsf{CMBT.Sign}_0(\mathsf{pp}, \mathsf{sk}, b, T')$
                                          $(X_0, X_1, \tilde{X}) \coloneqq \mathsf{pp}$
                                          $((x_0, y_0), (x_1, y_1), (\tilde{x}, \tilde{y})) \coloneqq \mathsf{sk}$
                                          $s \leftarrow\!\$\ \{0,1\}^\lambda$
                                          $S' \coloneqq \mathsf{H}_s(T', s)$
                                          $W' \coloneqq x_b T' + y_b S'$
                                          $\tilde{W}' \coloneqq \tilde{x} T' + \tilde{y} S'$
                                          $\pi \leftarrow \Pi_{\mathsf{DLEQOR2}}.\mathsf{Prove}((X_0, X_1, T', S', W'), (b, x_b, y_b))$
                                          $\tilde{\pi} \leftarrow \Pi_{\mathsf{DLEQ2}}.\mathsf{Prove}((\tilde{X}, T', S', \tilde{W}'), (\tilde{x}, \tilde{y}))$
                                          return $(s, W', \tilde{W}', \pi, \tilde{\pi})$

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

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

Fig. 22. Token issuance for CMBT (Construction 6).