Publicly verifiable anonymous tokens with private metadata bit

Fabrice Benhamouda[1], Tancrède Lepoint[2], Michele Orrù[3], and Mariana Raykova[4]

[1] Algorand Foundation fabrice.benhamouda@gmail.com
[2] Independent researcher, crypto@tancre.de
[3] UC Berkeley, michele.orru@berkeley.edu
[4] Google, marianar@google.com
[5]

Abstract. We present a new construction for publicly verifiable anonymous tokens with private metadata. This primitive enables an issuer to generate an anonymous authentication token for a user while embedding a single private metadata bit. The token can be publicly verified, while the value of the private metadata is only accessible to the party holding the secret issuing key and remains hidden to any other party, even to the user. The security properties of this primitive also include unforgeability, which guarantees that only the user can generate new valid tokens, and unlinkability that guarantees that tokens issued with the same private metadata bit are indistinguishable. Our anonymous tokens scheme builds on the top of blind Schnorr signatures. We analyze its security in the algebraic group model and prove its security under the modified ROS assumption, one-more discrete logarithm, and decisional Diffie-Hellman assumptions.

1 Introduction

The use of user identity on the Internet has two different connotations when viewed through the lenses of privacy and security. On the one hand, users should have the right to remain anonymous and still be able to access and use many sites and services on the Internet. On the other hand, authenticating user identity is a major tool for fraud protection that relies on trust bestowed upon users based on their long-term behavior in the ecosystem. This creates a seeming tension between scenarios where we want to both protect the user privacy and at the same time maintain important trust signals used for fraud detection. Anonymous credentials [Cha82, CL01] provide a mechanism to reconcile this tension enabling a trusted issuer to generate a credential that carries only a limited amount of trust signals about a user but does not reveal the user identity. Many settings facing the need of anonymous authentication need to support large volumes of traffic and thus the agility and efficiency of the authentication solution become crucial. That is why simple one-time use authentication tokens often become the tool of choice in such settings. For example, Privacy Pass [DGS+18] is an anonymous token based on verifiable oblivious pseudorandom function (VOPRF) that has been used by Cloudflare to gate access to content delivery networks (CDNs). Chrome has recently released an API for origin trial [Chr19] which provides an extended functionality where the anonymous token also carries an additional private metadata bit. This API aims to provide a tool that can recover trust signals in a privacy-preserving manner in a world without third party cookies while still preventing spam learning. In particular, trust tokens can be issued to users when they are in first party context, i.e., users are logged in, and when the issuer has richer information about the users to make trust determination. Consequently users can redeem these tokens when visiting different sites or services without being logged in and can prove this way that they are trustworthy while preserving their anonymity.

Both of the above examples rely on tokens that have secret key verification. Such functionality suffices in settings where the issuer and the verifier are the same party. Having one central verifier facilitates detection of double spending where the same single-use token is presented multiple times for authentication. However, restricting the verification to a single designated verifier comes with limitations for large heterogeneous systems where many parties could benefit from the ability to verify user identity, but not every party can

afford to set up a token issuing authority or has sufficient first party context from which to derive trust signal for users.

Anonymous tokens that provide public verifiability could open the doors for supporting such broader systems, where multiple verifiers can rely on a single authority. The approaches to double spending fraud detection in such systems still need to rely on centralized mechanisms provided by the issuing authority which often will not be real time. To facilitate such functionality, it is beneficial to enable the issuing authority to embed limited trust signals in the tokens that can be used in the fraud detection mechanisms. These trust signals should not be available to the adversary, in order to prevent it from adapting its behavior based on the change of this trust signal. This can be achieved by introducing the notion private metadata for the publicly verifiable tokens, which is data that the issuer can embed at token issuance and later read it off the anonymous token. But this metadata remains hidden from any other party including the user to which the token was issued. The interest in constructions of anonymous tokens with the above properties has been highlighted in an issue filed with the RFC draft of the IETF Privacy Pass Working Group [Gro20].

1.1 Contributions

In this paper, we present a public verifiability token scheme with private metadata bit. This functionality enables an issuer to issue a trust token to a client while attaching one additional bit of metadata. The public verifiability enables anyone to verify the validity of the token. The private metadata bit is accessible only to the issuer. Our construction provides *unforgeability*, which guarantees that only the holder of the secret key can generate valid tokens. It also has an *unlinkability* property which prevents linking tokens to user identity. In the context of the private metadata, this property guarantees tokens issued to the same or different users with the same private metadata bit are indistinguishable. Finally, the construction has *privacy for the metadata*, which guarantees that the value of the private metadata bit remains hidden to any party that does not have the issuing secret key including the client to whom the token was issued.

We present two constructions: a basic one and a "clause" version. For both constructions, security is proven in the random oracle and the algebraic group model. The algebraic group model [BV98, PV05] (AGM) restricts adversaries to be algebraic, that is: every time the adversary output a group element, it must also output a representation of the element as a linear combination of the previously seen group elements. The algebraic group model has been studied as an independent notion in [FKL18].

Both our constructions assume the hardness of the Decisional Diffie-Hellman assumption (DDH) and of the One-More Discrete Logarithm assumption (OMDL). In addition, to prove unforgeability (that is, the inability for a user to create tokens with invalid metadata bits or create more tokens than the number of interactions with the issuer), our basic construction relies on the regular ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) assumption [Sch01a, FPS20], while our clause version relies on the modified ROS assumption [FPS20].

Recent works by Wagner [Wag02] and Benhamouda et al. [BLL+21] show subexponential-time and polynomial-time attacks against the regular ROS assumption. This makes our basic construction insecure in most practical settings. As such, we see our basic construction as a stepping stone to better understand our clause version that rely on the modified ROS assumption for which no such attacks are known.

1.2 Overview

In this section, we overview the main technical ideas of our construction and put them on context of other related constructions.

A classical example of publicly verifiable unlinkable tokens are blind signatures [Cha82, CP93, Oka93, PS00, Sch01b, Sch06, FPS20, FHS15, Bol03], which on their own provide unlinkability and unforgeability but do not allow the signer to embed any private metadata. There are constructions such as partially blind signatures [Fis06, SC12, BPV12] which enable the signer to contribute part of the signed message and this can be considered as public metadata, but these constructions cannot provide privacy for the metadata. The anonymous tokens construction of Kreuter et al. [KLOR20] provides the private metadata bit property and unlinkability but works in the secret key setting for verification. This construction together with the Blind

Schnorr signatures in the algebraic group model of Fuchsbauer et al. [FPS20] will be the starting points for our publicly verifiable tokens with private metadata bit.

Secret key anonymous tokens [KLOR20] extend the idea of using verifiable oblivious pseudorandom functions (VOPRFs) as anonymous credentials of Privacy Pass [DGS+18]. Privacy Pass uses a cyclic group $\mathbb{G}$ of prime order p. (We use the additive notation for cyclic groups in this paper.) The Privacy Pass tokens are of the form $(t, x\mathsf{H}(t))$ where t is sampled at random, H is a hash function outputting group elements and modeled as a random oracle, and x is a secret key. The issuance of the anonymous token leverages the oblivious evaluation of the PRF $x\mathsf{H}(t)$ where the client provides $r\mathsf{H}(t)$ and obtains back $xr\mathsf{H}(t)$ together with a discrete log equality (DLEQ) proof that the key x used in the evaluation is the same as the one committed in a public parameter $X = xG$. The idea of Kreuter et al. [KLOR20] for adding metadata bit is to use two sets of parameters x_0, x_1 where tokens issued with x_b carry private metadata bit b. The verifiability property for the client is achieved by extending the DLEQ proof to an OR proof of discrete log equalities [CDS94] which enables the issuer to prove that the token is issued with one of the two committed keys without revealing which one. In order to achieve privacy of the metadata bit when the same value t is used twice, this idea is further extended with additional rerandomization on the issuer side.

Blind Schnorr signatures provide a way for the client to *blind* the message that the issuer signs in a discrete-log based signature in a way that the client can derive the correct signature only for the initially blinded message. The starting point is a discrete-log signature scheme, where given a public parameter $X = xG$ the signature of a message m is of the form (K, r) where $K = kG$ for a randomly chosen k and $r = k + \mathsf{H}_e(K, m)x$ and the verification checks that $rG = K + \mathsf{H}_e(R, m)X$ where $\mathsf{H}_e : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle. The idea of the blind signature scheme is to have the client compute a rerandomized version $K' = K + \alpha G + \beta X$ of K that is sent to the issuer together with a rerandomized version of the challenge $e = H_e(K', m) + \beta$. At that point the signer can produce a signature using the blinded values computing $r = k + ex$. Subsequently the client verifies the signature (K, r) (i.e., verifies that $rG = K + eX$). If the verification passes, the client computes an unblinded signature (K', r') setting $r' = r + \alpha$. The verification for the final signature holds since $r'G = rG + \alpha G = (k + ex)G + \alpha G = rG + H(K', m)xG + \beta xG + \alpha G = K' + H(K', m)X$.

The idea of our public key token construction with private metadata is to apply the idea from the secret key construction above of having two sets of parameters to enable the private metadata bit, to the blind Schnorr signature construction. However, since in the public verifiability setting anyone can verify the correctness of the token we cannot use only one of the set of parameters for each token. Instead our construction will effectively be issuing two publicly verifiable tokens that verify and we will be embedding the private metadata bit in a way that only the issuer can recover it. To expand, for each token issuance, we will be deriving a new pair of public key parameters for the Schnorr blind signatures, where one of them will be "valid" and the other will be "invalid" depending on the private metadata bit. Only the valid one of these two sets of public parameters will correspond to the secret key x_b where b is the value of the private metadata bit. The final token will consist of two verifiable tokens under the two sets of parameters. The private metadata bit will be readable by identifying which public parameters are valid, which can be done only by the issuer.

In the Schnorr blind signature scheme, the proof of correct issuance is achieved with a sigma protocol. The issuer will be able to give such a proof only for the signature under the valid set of parameters. That is why the issuer will use an OR-proof of sigma protocols for the two parallel executions to generate two verifiable blind signatures under the two sets of parameters. Public keys or parameters of classical Schnorr signatures just consist in a single group element X and hence are always valid. In order to allow for both valid and invalid public parameters, we use ideas from the Diffie-Hellman signatures of Katz and Wang [KW03] which have as a public key a vector $\mathsf{pk} := (X = xG, Y = xH)$ and effectively run two discrete-log signatures in parallel using the two elements in the public key and the same challenge. In particular the signature is of the form (K, C, s) where $K = kG$, $C = kH$ for a random k, and $r = k + \mathsf{H}(\mathsf{pk}, K, C, m)x$. The verification check that $rG = K + H(\mathsf{pk}, K, C, m)\, X$ and $sH = C + H(\mathsf{PK}, K, C, m)\, Y$. Note that only a public key that is a DDH tuple is a valid public key for this scheme. An invalid key will be a pair of random elements.

We can turn the above DH signature into a blind signature, following the same technique behind Blind Schnorr signatures. However, this so far has not achieved our goal to restrict the issuer's capability to be able to embed only one hidden bit value, i.e., we want to guarantee that for each token execution the issuer can create at most one set of valid public parameters for the DH signatures. We achieve this as follows: our token scheme will have a public key $X_0 = x_0 G$, $X_1 = x_1 G$ and a secret signing key (x_0, x_1). During signing the issuer generates $Y = x_b H_b$ where H_0, H_1 are generated by a random oracle, and b is the private metadata bit for the signature. The issuer uses (X_0, Y) and (X_1, Y) as public parameters for the two parallel signatures that it will be issuing. In the same way of a OR sigma protocol, the signer can honestly sign only using (X_b, Y), and must simulate the proof under key (X_{1-b}, Y). Thus, for its first message the issuer will generate honestly $[K_b; C_b] = [k_b G; k_b H_b]$ with k_b being sampled at random and will simulate the values $[K_{1-b}; C_{1-b}]$ as follows:

$$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} := r_{1-b} \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix},$$

where r_{1-b} and e_{1-b} are random.

Next, we need to adapt the ideas from the blind Schnorr signatures in order for the client to rerandomize the commitments $[K_0, C_0]$ and $[K_1, C_1]$ and create the challenge for the issuer. The client does that by computing

$$\begin{bmatrix} K'_i \\ C'_i \end{bmatrix} = \begin{bmatrix} K_i \\ \rho C_i \end{bmatrix} + \alpha_i \cdot \begin{bmatrix} G \\ \rho H_i \end{bmatrix} + \beta_i \cdot \begin{bmatrix} X_i \\ \rho Y \end{bmatrix},$$

for $i = 0, 1$ where $\rho, \alpha_0, \alpha_1, \beta_0, \beta_1 \leftarrow_\$ \mathbb{Z}_p$. Consequently the client also computes the challenge as $e := \mathsf{H}(\rho Y, \rho H_0, \rho H_1, K'_0, K'_1, C'_0, C'_1, m) - \beta_0 - \beta_1$.

The issuer provides two verifying DH signatures for m with public parameters (X_0, Y) and (X_1, Y) as follows: it sets $e_b = e - e_{1-b}$ and $r_b = k_b - e_b x_b$. The client verifies

$$\begin{bmatrix} K_i \\ C_i \end{bmatrix} = r_i \cdot \begin{bmatrix} G \\ H_i \end{bmatrix} + e_i \cdot \begin{bmatrix} X_i \\ Y \end{bmatrix},$$

for $i = 0, 1$ and $e = e_0 + e_1$, and if these hold outputs a signature $(H'_0 = \rho H_0, H'_1 = \rho H_1, Y' = \rho Y, (e_i + \beta_i, r_i + \alpha_i)_{i=0,1})$.

The public key verification of a signature $(H'_0, H'_1, Y', (e_i + \beta_i, r_i + \alpha_i)_{i=0,1})$ of a message m in the above scheme consists in computing:

$$\begin{bmatrix} \bar{K}_i \\ \bar{C}_i \end{bmatrix} := r'_i \cdot \begin{bmatrix} G \\ H'_i \end{bmatrix} + e'_i \cdot \begin{bmatrix} X_i \\ Y' \end{bmatrix},$$

and checking that $e_0 + e_1 = \mathsf{H}(Y', \bar{K}_0, \bar{K}_1, \bar{C}_0, \bar{C}_1, m)$.

Finally, the issuer can read the private metadata bit associated with a signature $(H'_0, H'_1, Y', (e_i + \beta_i, r_i + \alpha_i)_{i=0,1})$ testing for the bit b such that $Y' = x_b H'_b$. If the equation does not hold for either value of b (or, it holds for both), then it returns $\perp$.

The clause version. Achieving unforgeability in the above construction similarly to the Blind Schnorr signatures scheme can be proven assuming the hardness of the ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) problem [Sch01a, FPS20] in the algebraic group model (AGM). Recent works of Wagner [Wag02] and Benhamouda et al. [BLL+21] showed subexponential-time algorithms solving the ROS problem for any dimension and a polynomial-time algorithm when the dimension is logarithmic in the domain size of the elements. Thus we extend the above construction into a "clause" construction using techniques introduced by Fuchsbauer et al. [FPS20], which achieves unforgeability under a potentially harder assumption, the modified ROS assumption [FPS20]. Current attack techniques [Wag02, BLL+21] do

not apply to modified ROS and Fuchsbauer et al. [FPS20] provide evidence that the hardness of this problem is different in some significant ways from the hardness of the regular ROS assumption.

The ROS problem, parameterized by an integer ℓ, asks an adversary to find $\ell+1$ vectors $\boldsymbol{\rho}_i = (\rho_{i,j})_{j\in[\ell]}$ such that the system of $\ell+1$ equations $\sum_{j=1}^{\ell} \rho_{i,j} x_j = \mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_i)$ over ℓ variables $x_1, \ldots, x_\ell$ in $\mathbb{Z}_p$ has a solution, where $\mathsf{H}_{\mathsf{ROS}} : (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ is a random oracle. The modified ROS assumption differs from the regular ROS assumption as follows: the adversary needs to come up with a pair of vectors $\rho_i^k = (\rho_{i,j}^k)_{j\in[\ell]}$, $k = 0, 1$ and the linear system of equations that needs to be satisfied is $\sum_{j=1}^{\ell} \rho_{i,j}^{b_j} x_{j,b_j} = \mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_i^0, \boldsymbol{\rho}_i^1)$ where b_j is the output of a random oracle $\textsc{Select}(j, x_{j,0}, x_{j,1})$. There is also an additional requirement for the adversary's success which is that $\rho_i^{1-b_j} = 0$ for all $i \in [\ell+1], j \in [\ell]$.

The idea for our clause construction is a natural generalization similar to the modified ROS extension of the base ROS problem: there will be two sets of parameters of the public verifiability construction with private metadata described above. During token issuance the issuer and the client will be running two executions in parallel of the blind signing using the two sets of parameters and in its last message the issuer will choose and complete only one of the executions.

Security intuition. The unforgeability property of our construction is proven in the AGM model under the modified ROS assumption. An unforgeability adversary can succeed in three main ways: generating a valid token that allows reading both possible values of the private metadata bit or a token from which no private metadata value can be read, or generating $\ell+1$ valid tokens that have a private metadata bit b while the adversary has been issued less than ℓ tokens with this bit.

The first type of misbehavior is ruled out in the AGM model by showing that if the adversary manages to generates non-zero group elements H_0, H_1, and $Y = x_0 H_0 = x_1 H_1$, then in the algebraic representation of Y, under the one-more discrete log (OMDL) assumption, there must be a multiple of $x_0 x_1$, which is impossible. An adversary producing a valid token with no private metadata bit can be used to generate a forgery OR proof for knowledge of one of two discrete logs since the token verifies and yet $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$. The analysis of the remaining case when the adversary generates more valid signatures with the same private metadata bit than what it queried for, uses the AGM model to derive a sequence of linear equations from the verification equation for each valid signature, which induce a system of linear equations modulo p. If the system has a solution, its solution allows to build an adversary for the modified ROS assumption. Otherwise, if the linear system does not have a solution, we can reduce security to the OMDL assumption: the reduction uses the discrete logarithm oracle from OMDL to simulate the responses to token queries and uses the system of equations from the verification of the $\ell+1$ tokens to extract another discrete logarithm with respect to the scheme secret key.

For the unlinkability property of the construction we provide a new definition that reflects the presence of the private metadata which provides one bit of information to the issuer. The goal of this definition is to capture the intuition that the issuer cannot distinguish any two tokens issued with same private metadata bit (hence any other party cannot distinguish any two tokens). The definition uses a security game where the adversary is given oracle access to the user algorithms and can run as many sessions as it wants. It also can request the final unblinded tokens in any execution. Finally the adversary has access to a challenge oracle where it provides two execution session indices that use the same private metadata bit and for which it has not requested the issued tokens, and the challenger returns the tokens for those sessions in a random order (but using the same order across all oracle calls). The goal of the adversary is to guess the order bit used by the challenge oracle. The unlinkability argument observes that the only part of the client's computation that depends on the first messages received from the issuer is the hash for the challenge. Using the programmability of the random oracle model used for the hash, we can eliminate this dependence and thus make the outputs of tokens generated with the same private metadata bit indistinguishable.

To prove the private metadata property of our scheme we leverage the AGM model and the programmable random oracle to construct tokens where both metadata bit values are set and the proofs of both sigma protocols are simulated. We show that such tokens are computationally indistinguishable from honest tokens

where either of the private metadata bit values is set, which demonstrates the indistinguishability of tokens with different metadata bit values.

Related work. The notion of single use anonymous tokens dates back to the work of Chaum [Cha82] introducing the concept of blind signatures, which has been widely used as a tool for building anonymous credentials. The security of that construction has been analyzed in different models by several follow-up works [CP93, Oka93, PS00, Sch01b, Sch06, FPS20]. Constructions of partially blind signatures [Fis06, SC12, BPV12] start to enable functionality that allows the issuer to embed some additional *public* information to the signed message. The work of Baldimtsi et al. [BL13] presents a blind signature construction with attributes where the user can provide a zero-knowledge proof for possession of certain attributes and the issuer can include a re-randomized version of the committed attributes in the signature. Tsang et al. [TAKS07] present a construction for blacklistable anonymous credentials using bilinear maps, where the issuer defines a list of blocked users and during issuance the user obtains an authentication token only if she is not on the list; hence she finds out whether she has been added to the list.

Anonymous tokens constructions exist also in the secret key setting. Abdalla et al. [ANN06] introduced a secret key analog to blind signatures, which is called blind message authentication codes (MACs). 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] introduced the notion of an anonymous token called Privacy Pass, which has a similar private key functionality that uses verifiable oblivious pseudorandom functions (VOPRFs) [JKK14]. A recent paper of Kreuter et al. [KLOR20] presents secret key anonymous tokens that additionally allow the issuer to embed a private metadata bit in each token that remains hidden to the user and any other party that does not have the secret key.

1.3 Roadmap

After some preliminaries in Section 2, we define anonymous tokens with private metadata bit and public verification in Section 3. We show our basic construction in Section 4 and prove its security. This construction relies on the ROS assumption and hence is not secure. Finally, in Section 5, we describe its clause version and proves its security in the AGM and random oracle model under the DDH, the OMDL, and the modified ROS assumption.

2 Preliminaries

2.1 Notation

For an integer n, we denote with $[n]$ the integer interval $\{0, \ldots, n-1\}$. We denote vectors in bold: for a vector $\mathbf{a}$, we denote with a_i its i-th element. A function $\mu : \mathbb{N} \to [0,1]$ is negligible (denoted $\mu = \mathsf{negl}$) if for all $c \in \mathbb{N}$ there exists $\lambda_c \in \mathbb{N}$ such that $\mu(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$. When sampling the value x uniformly at random from the set S, we write $x \leftarrow\!\!\$\, S$. When sampling the value x from the probabilistic algorithm M, we write $x \leftarrow \mathsf{M}$. We use $:=$ to denote assignment.

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 $\Gamma = (\mathbb{G}, p, G)$ of a group $\mathbb{G}$ of prime order p. For simplicity, we will assume that the prime p is of length λ. We employ additive notation for groups.

2.2 Assumptions

The ROS problem. Let p be a prime number and $\mathsf{H}_{\mathsf{ROS}}$ a random oracle with range in $\mathbb{Z}_p$. The ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) problem [Sch01a, FPS20] for ℓ dimensions, displayed in Figure 1, asks to find $\ell+1$ vectors $\boldsymbol{\rho}_i = (\rho_{i,j})_{j\in[\ell]}$ (for $i \in [\ell+1]$), and $\ell+1$

Game $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$	Oracle $\mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}, \mathsf{aux})$
$p \leftarrow \mathsf{Pgen}(1^\lambda)$ $\mathsf{T} := [\,]$ $\left((\boldsymbol{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}, (c_j)_{j\in[\ell]}\right) \leftarrow \mathsf{A}^{\mathsf{H}_{\mathsf{ROS}}}(p)$ **return** $\big(\ \forall i \neq j \in [\ell+1],\quad (\boldsymbol{\rho}_i, \mathsf{aux}_i) \neq (\boldsymbol{\rho}_j, \mathsf{aux}_j)$ $\quad \wedge\ \forall i \in [\ell+1],\quad \sum_{j\in[\ell]} c_j \rho_{i,j} = \mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_i, \mathsf{aux}_i)\big)$	**if** $\mathsf{T}[\boldsymbol{\rho}, \mathsf{aux}] = \perp$ **then** $\quad \mathsf{T}[\boldsymbol{\rho}, \mathsf{aux}] \leftarrow\!\!\$\ \mathbb{Z}_p$ **return** $\mathsf{T}[\boldsymbol{\rho}, \mathsf{aux}]$

Fig. 1. The $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ game.

Game $\mathrm{mROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$	Oracle $\mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \mathsf{aux})$

$p \leftarrow \mathsf{Pgen}(1^\lambda)$
$\mathsf{T} := [\,]$
$(\boldsymbol{\rho}_{i,0}, \boldsymbol{\rho}_{i,1}, \mathsf{aux}_i)_{i\in[\ell+1]} \leftarrow \mathsf{A}^{\mathsf{H}_{\mathsf{ROS}}}(p)$
return $\big(\ \forall i \neq j \in [\ell+1],\quad (\boldsymbol{\rho}_i, \mathsf{aux}_i) \neq (\boldsymbol{\rho}_j, \mathsf{aux}_j)$
$\quad \wedge\ \forall i \in [\ell+1],\quad \sum_{j\in[\ell]} c_j \rho_{i,b_j,j} = \mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_i, \mathsf{aux}_i)$
$\quad \wedge\ \forall i \in [\ell+1], \forall j \in [\ell],\quad \rho_{i,1-b_j,j} = 0\big)$
	if $\mathsf{T}[\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \mathsf{aux}] = \perp$ **then**
	$\quad \mathsf{T}[\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \mathsf{aux}] \leftarrow\!\!\$\ \mathbb{Z}_p$
	return $\mathsf{T}[\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \mathsf{aux}]$
	
	Oracle $\textsc{Select}(j, c'_0, c'_1)$
	// Must be queried once $\forall j \in [\ell]$
	$b_j \leftarrow\!\!\$\ \{0,1\};\quad c_j := c'_{b_j}$
	return b_j

Fig. 2. The $\mathrm{mROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ game.

bit strings $\mathsf{aux}_i \in \{0,1\}^*$ (for $i \in [\ell+1]$), as well as a vector $\mathbf{c} = (c_j)_{j\in[\ell]}$ such that:

$$\mathsf{H}_{\mathsf{ROS}}(\boldsymbol{\rho}_i, \mathsf{aux}_i) = \sum_{j\in[\ell]} c_j \rho_{i,j} \qquad \text{for all } i \in [\ell+1].$$

Formally, we define ROS using the game $\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ in Fig. 1.

Definition 1. *Let* Pgen *be a prime-number generator. The ROS problem in dimension ℓ is* hard *for* Pgen *if for all PPT adversaries:*

$$\mathsf{Adv}^{\mathrm{ros}}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) := \Pr\left[\mathrm{ROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

The modified ROS problem. The modified ROS problem was introduced in [FPS20]. We refer to that paper for intuition for its hardness compared to ROS problem. In short, the ROS problem requires the adversary to query the hash oracle with two vectors $\boldsymbol{\rho}_{i,0}$ and $\boldsymbol{\rho}_{i,1}$ instead of just one $\boldsymbol{\rho}_i$. The vector $\mathbf{c}$ is selected by the adversary querying a second oracle $\textsc{Select}(j, c_{j,0}, c_{j,1})$ for each index $j \in [\ell]$. This oracle picks a random bit $b_j \leftarrow\!\!\$\ \{0,1\}$ and sets $c_j := c_{j,b_j}$. Note that if there was no other constraint on the solution, modified ROS would actually not be harder than ROS as an adversary could just query $\mathsf{H}_{\mathsf{ROS}}$ with $\boldsymbol{\rho}_{i,0} = \boldsymbol{\rho}_{i,1} = \boldsymbol{\rho}_i$, and then make $\textsc{Select}$ queries with $c_{j,0} = c_{j,1} = c_j$. What makes modified ROS problem potentially harder is that the solution is required to be such that $\rho_{i,1-b_j,j} = 0$ for all $j \in [\ell]$. In other words, the non-selected components of $\boldsymbol{\rho}_{i,b}$ are required to be zero.

Formally, we define modified ROS using the game $\mathrm{mROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda)$ in Fig. 2.

Definition 2. *Let* Pgen *be a prime-number generator. The* modified *ROS problem in dimension ℓ is* hard *for* GrGen *if for all PPT adversaries:*

$$\mathsf{Adv}^{\mathrm{mros}}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) := \Pr\left[\mathrm{mROS}_{\mathsf{Pgen},\mathsf{A},\ell}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

Game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$	Oracle $\textsc{Target}()$	Oracle $\textsc{Help}(X)$
$(\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$i := i + 1$	$q := q + 1$
$\mathbf{x} \leftarrow\!\!\$\ \mathbb{Z}_p^{\ell+1};\ i := 0;\ q := 0$	**if** $i > \ell + 1$ **then return** $\bot$	**if** $q > \ell$ **then return** $\bot$
$\mathbf{y} \leftarrow \mathsf{A}^{\textsc{Target},\textsc{Help}}(p)$	**return** $x_{i-1}G$	**return** $\textsc{Dlog}(X)$
return $\mathbf{y} = \mathbf{x}$		

Fig. 3. The $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{A},\ell}(\lambda)$ game.

OMDL. The one-more discrete logarithm assumption (OMDL) problem [BNPS03], displayed in Fig. 3, asks a PPT adversary A to compute the DL of $\ell + 1$ challenges, with the help of a $\textsc{Dlog}$ oracle (denoted $\textsc{Help}$) that returns the discrete log of up to ℓ challenges.

Definition 3. *Let* GrGen *be a group generator. The one-more discrete logarithm problem with* $\ell+1$ *challenges is* hard *for* GrGen *if for all PPT adversaries:*

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

DDH. We recall the Decisional Diffie-Hellman problem [DH76], which given group parameters $(\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ asks an adversary to distinguish a DDH tuple $(xG, yG, xyG) \in \mathbb{G}^3$ (game $\mathrm{DDH}^0_{\mathsf{GrGen},\mathsf{A}}(\lambda)$) from a uniformly random tuple in $\mathbb{G}^3$ (game $\mathrm{DDH}^1_{\mathsf{GrGen},\mathsf{A}}(\lambda)$).

Definition 4. *Let* GrGen *be a group generator. The DDH problem is hard for* GrGen *if for all PPT adversaries:*

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

Algebraic group model. Most of our proofs will be in the so-called *algebraic group model* [BV98, PV05, FKL18]. In this model, we consider only algebraic adversaries $\mathsf{A}_{\mathrm{alg}}$ that, for any output group element Z, also output a vector $\boldsymbol{\chi}$ that expresses Z as a linear combination of all previously received group elements. If so far $\mathsf{A}_{\mathrm{alg}}$ has received $G, X_1, \ldots, X_\ell$ from the challenger, and it outputs a group element $Z \in \mathbb{G}$, then we write $Z_{[\boldsymbol{\chi}]}$ to explicit also the coefficients $\boldsymbol{\chi}$ of the linear combination, where $\chi_0 G + \sum_{i=1}^{\ell} \chi_i X_i = Z$. We abuse notation and often write $\boldsymbol{\chi}$ as a polynomial $\boldsymbol{\chi} = \chi_0 + \sum_{i=1}^{\ell} \chi_i \bar{x}_i \in \mathbb{Z}_p[\bar{x}_1, \ldots, \bar{x}_\ell]$. The bar over $\bar{x}_i$ is to differentiate the indeterminate $\bar{x}_i$ of the polynomials from the actual discrete logarithms $x_i \in \mathbb{Z}_p$ of $X_i \in \mathbb{G}$ (i.e., $X_i = x_i G$).

3 Anonymous Tokens

Anonymous tokens are single-use anonymous credentials that can be used for privacy-preserving ad-metrics and audience measurements. In this section, we describe the functionality that they provide, and describe the security properties that they should guarantee.

3.1 Public-key anonymous tokens

An anonymous token AT 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;

– $\sigma \leftarrow \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b)\rangle$, the token issuance protocol (aka signing protocol), that involves interactive algorithms $\mathsf{AT.Usr}$ (run by the user) with input a tag $t \in \{0,1\}^*$, and $\mathsf{AT.Sign}$ (run by the issuer) with input the private key sk and a private metadata bit b. At the end of the interaction, the issuer outputs nothing, while the user outputs σ, or $\perp$.
– $bool \leftarrow \mathsf{AT.Ver}(\mathsf{pp}, t, \sigma)$, the verification algorithm that takes as input the public parameters pp and a token (t, σ). It returns a boolean indicating if the token is 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 $\mathsf{ind} \in \{\perp, 0, 1\}$, which is either the private metadata bit, or $\perp$.

Throughout the rest of this paper, we assume that AT has a two-round signing protocol initiated by the server. Thus, for simplicity, we split the signing algorithms ($\mathsf{AT.Sign}$ and $\mathsf{AT.Usr}$) into non-interactive algorithms that take as input a message, and the partial state (if any). They will return the next message together with the updated state st_i. Concretely, the signing protocol will be composed of the following (non-interactive) algorithms:

– $(\mathsf{resp}_S, \mathsf{st}_S) \leftarrow \mathsf{AT.Sign}_0(\mathsf{sk}, b)$;
– $(\mathsf{resp}_U, \mathsf{st}_U) \leftarrow \mathsf{AT.Usr}_0(\mathsf{pp}, t, \mathsf{resp}_U)$;
– $\mathsf{resp}_S \leftarrow \mathsf{AT.Sign}_1(\mathsf{st}_S, \mathsf{resp}_U)$;
– $\sigma \leftarrow \mathsf{AT.Usr}_1(\mathsf{st}_U, \mathsf{resp}_S)$

Definition 5 (Correctness). *An anonymous token scheme* AT *is correct if* $\forall t \in \{0,1\}^*,\ b \in \{0,1\},\ (\mathit{crs}, \mathit{td}) \in [\mathsf{AT.Setup}(1^\lambda)], (\mathsf{pp}, \mathsf{sk}) \in [\mathsf{AT.KeyGen}(\mathit{crs})]$:

$$\Pr[\mathsf{AT.Ver}(\mathsf{pp}, t, \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b)\rangle)] > 1 - \mathsf{negl}(\lambda)\ , \tag{1}$$

$$\Pr[\mathsf{AT.ReadBit}(\mathsf{sk}, t, \langle \mathsf{AT.Usr}(\mathsf{pp}, t), \mathsf{AT.Sign}(\mathsf{sk}, b)\rangle) = b] > 1 - \mathsf{negl}(\lambda)\ . \tag{2}$$

In addition to correctness an anonymous token scheme satisfies the following security properties: unforgeability, unlinkability, and privacy of the metadata bit.

3.2 Unforgeability

The first security property that we want from an anonymous token is *unforgeability*, which guarantees that a party that does not have the secret key cannot generate more valid anonymous tokens than issued. In particular an adversary who obtains ℓ valid tokens for each private metadata bit should not be able to generate $\ell + 1$ valid tokens for the same private metadata bit. To rule out trivial attacks, all $\ell + 1$ tags must be different. Furthermore, an adversary should not be able to generate a token for which no private metadata bit can be extracted. (This latter case is captured by the winning condition $s_0 + s_1 \neq m$ in Fig. 4). Note that generating a token from which both possible metadata bits can be read is already captured as forgery by the definition.

Definition 6 (One-more unforgeability). *An anonymous token scheme* AT *is* one-more unforgeable*, if for any PPT adversary* A *and any* $\ell > 0$*:*

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

where $\mathrm{OMUF}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ *is described in Figure 4.*

3.3 Unlinkability

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

We allow the adversary to choose arbitrarily the public parameters pp and do not require they are generated honestly, which makes the definition stronger.

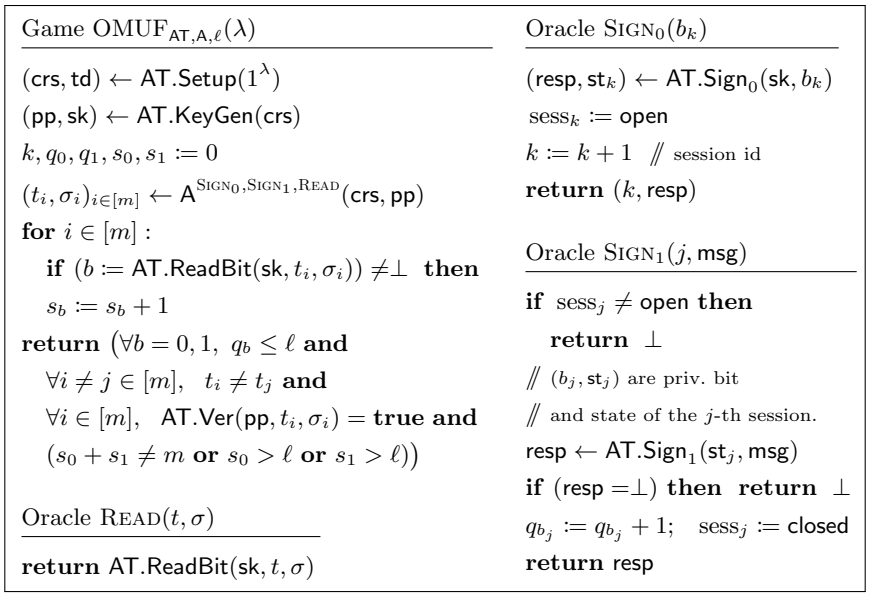

Fig. 4. One-more unforgeability game for the anonymous token scheme AT.

Definition 7 (Unlinkability). *An anonymous token scheme* AT *is* unlinkable*, if for any* $\ell > 0$ *and any PPT adversary* A *there exists an extractor* Ext*:*

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

where $\mathrm{UNLINK}^\beta_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ *is defined in Figure 5.*

Remark 1. We note that in the security experiment $\mathrm{UNLINK}_{\mathsf{AT},\mathsf{A},\ell}(\lambda)$ selects the message for the user uniformly at random. This is inherited by previous formalizations for anonymous tokens [DGS+18, KLOR20] and reported in the same way for uniformity. We note, however, that in the proof for Theorem 3 this is never formally required.

3.4 Private metadata bit

The last security property that we define concerns guarantees that the user to whom a token was issued, or any other party, cannot learn any information about the private metadata bit associated with that token. Intuitively, our definition guarantees that an adversary who can obtain tokens for arbitrary tags and metadata bits, and an arbitrary number of tokens for messages with the fixed challenge bit, cannot guess the challenge bit with a probability non-negligibly better than 1/2.

Definition 8 (Private metadata bit). *An anonymous token scheme* AT *provides* private metadata bit *if for all adversary* $\mathsf{A} \in$ *PPT:*

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

where $\mathrm{PMB}^\beta_{\mathsf{AT},\mathsf{A}}(\lambda)$ *is illustrated in Figure 6.*

In the formalization of the game, the adversary interacts with the issuer in up to m sessions, in two phases. In the first phase, the adversary also has access to a AT.ReadBit oracle. Then, in the second phase, the adversary loses access to the AT.ReadBit oracle and instead gets access to a challenge oracle that simulates the issuer with a random bit β. The goal of the adversary is to guess β.

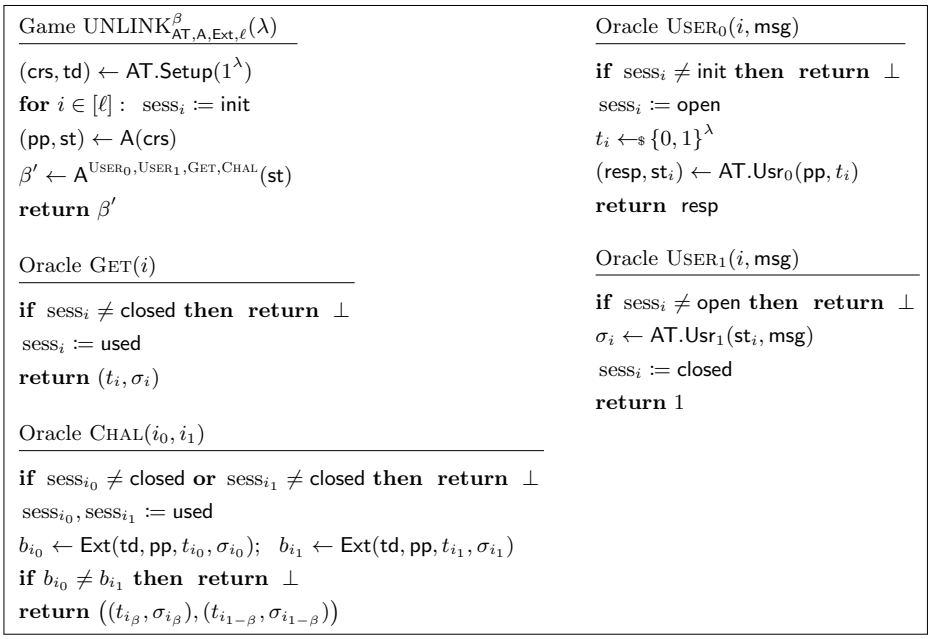

Fig. 5. Unlinkability game for the anonymous token scheme AT.

Note that it is not possible to give access to both the AT.ReadBit oracle and the challenge oracle at the same time, otherwise the adversary could easily find out β by querying the token resulting from a challenge interaction in AT.ReadBit. Due to unlinkability, it seems complex to restrict the AT.ReadBit oracle to not answer tokens resulting from a challenge interaction (as done in more classical security notions such as IND-CCA2).

4 Our construction

Our base construction Σ is described in Fig. 7. Ideas underlying this construction were presented in Section 1.2. Below, we prove correctness of the construction, introduces an important lemma used to prove security, and then prove security of the construction. Concretely, we prove privacy of the metadata bit, unforgeability, and unlinkability.

4.1 Correctness

Lemma 1. *The scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) is a correct anonymous token scheme.*

Proof. We start by proving that an honest user interacting with an honest issuer outputs a valid token that passes the verification (AT.Ver). The protocol does not abort because, for $i = 0, 1$, if $i = b$:

$$k_b = r_b + e_b x_b, \text{ and } \begin{bmatrix} K_b \\ C_b \end{bmatrix} = k_b \begin{bmatrix} G \\ H_b \end{bmatrix} \implies \begin{bmatrix} K_b \\ C_b \end{bmatrix} = r_b \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix},$$

and if $i = 1 - b$:

$$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} := r_{1-b} \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}.$$

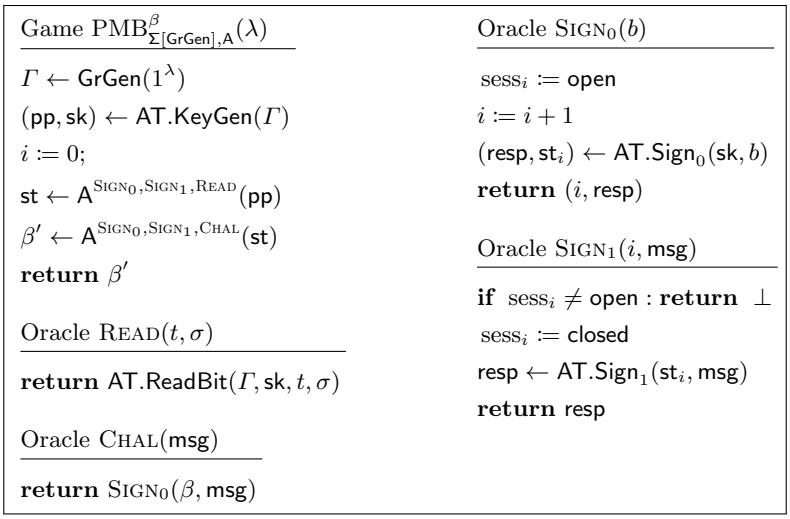

Fig. 6. Private metadata bit game.

Furthermore, we remark that for $i = 0, 1$:

$$\begin{aligned}\begin{bmatrix}\bar{K}_i\\ \bar{C}_i\end{bmatrix} &= r'_i \cdot \begin{bmatrix}G\\ H'_i\end{bmatrix} + e'_i \cdot \begin{bmatrix}X_i\\ Y'_i\end{bmatrix}\\ &= r_i \cdot \begin{bmatrix}G\\ H'_i\end{bmatrix} + \alpha_i \cdot \begin{bmatrix}G\\ H'_i\end{bmatrix} + e_i \cdot \begin{bmatrix}X_i\\ Y'\end{bmatrix} + \beta_i \cdot \begin{bmatrix}X_i\\ Y'\end{bmatrix}\\ &= \begin{bmatrix}K_i\\ \rho C_i\end{bmatrix} + \alpha_i \cdot \begin{bmatrix}G\\ H'_i\end{bmatrix} + \beta_i \cdot \begin{bmatrix}X_i\\ Y'\end{bmatrix} = \begin{bmatrix}K'_i\\ C'_i\end{bmatrix} \ .\end{aligned}$$

Hence, $\bar{e} = e'$ as the hash inputs are the same. We conclude by remarking that $e'_0 + e'_1 = e_0 + \beta_0 + e_1 + \beta_1 = e + \beta_0 + \beta_1 = e' = \bar{e}$.

We now prove that such an honestly generated token is so that the read bit algorithm ($\mathsf{AT.ReadBit}$) output the correct metadata bit. This is because:

$$Y' = \rho Y = \rho x_b H_b = x_b H'_b \ ,$$

and with overwhelming probability the private metadata bit b is unique. In fact:

$$Y = x_b H_b = x_{1-b} H_{1-b} \implies H_b = x_{1-b} x_b^{-1} H_{1-b},$$

and since H_0, H_1 are uniformly distributed in $\mathbb{G}$, this event happens with probability $1/p$.

4.2 A useful lemma

The following lemma lays the ground for the proofs of unforgeability and privacy of the metadata bit. Essentially, it states that it is difficult for an adversary A to distinguish a Diffie-Hellman oracle for X_0 or X_1 that acts in the group from one that acts within the algebraic representation. This holds even if A has at disposal a signing oracle.

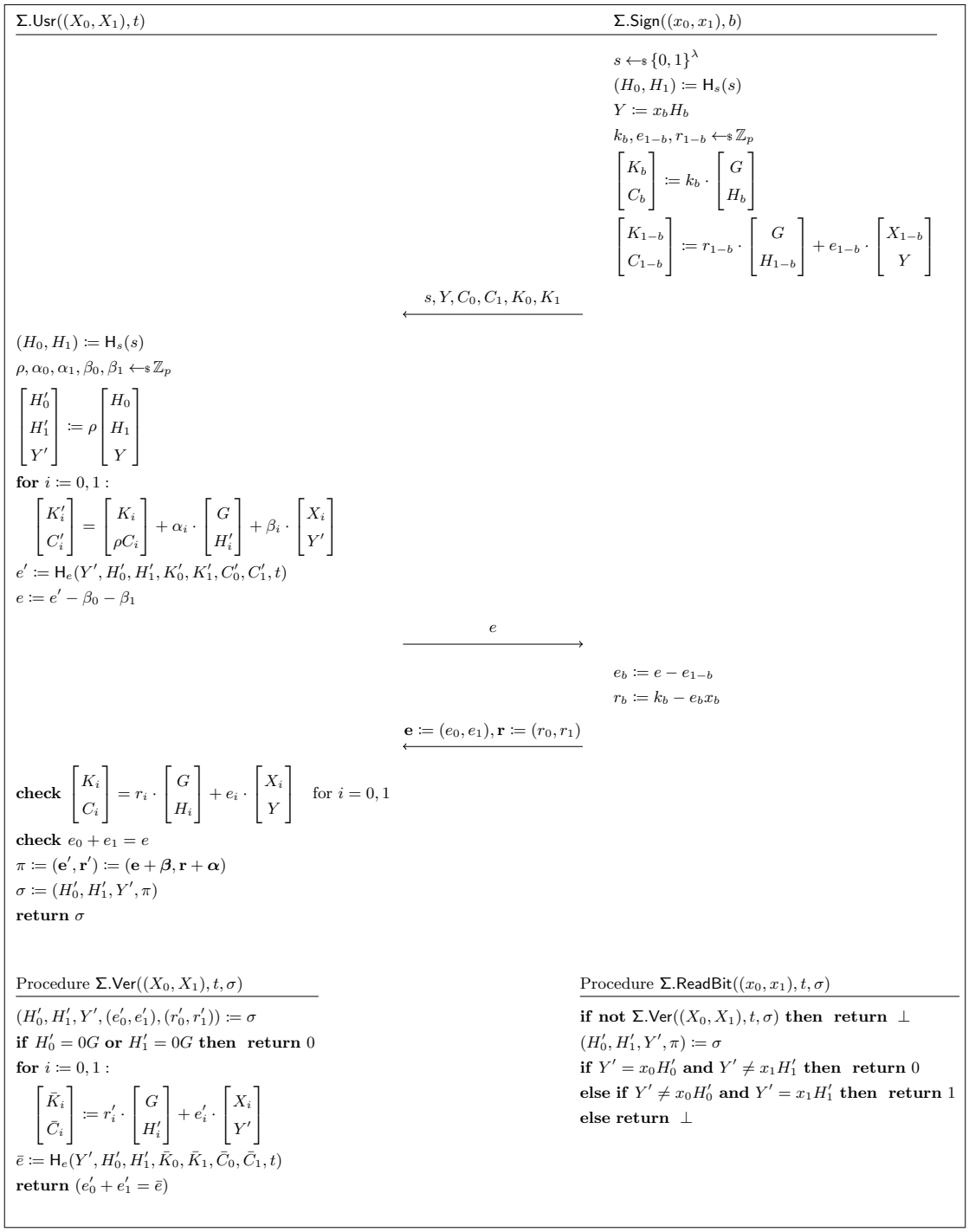

Fig. 7. The anonymous token protocol Σ.

Lemma 2. *Let the game* $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}}(\lambda)$ *be as defined in Fig. 8, where* $\mathrm{DH}^0_{\mathrm{chal}} = \mathrm{DH}$ *and* $\mathrm{DH}^1_{\mathrm{chal}} = \mathrm{DH}_{\mathrm{alg}}$. *For any algebraic adversary* A_{alg} *making at most* q *queries to the oracles* $\mathrm{DH}, \mathrm{DH}_{\mathrm{alg}}, \mathrm{DH}^{\beta}_{\mathrm{chal}}$, *the advantage*

Game $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}}(\lambda)$	Oracle $\mathrm{DH}(X,H,Y)$	Oracle $\mathrm{DH}_{\mathrm{alg}}(X,H_{[\boldsymbol{\eta}]},Y_{[\boldsymbol{\psi}]})$
$\Gamma := (\mathbb{G},p,G) \leftarrow \mathsf{GrGen}(1^{\lambda})$	**if** $X \neq X_0$ **or** $X \neq X_1$	**if** $X \neq X_0$ **or** $X \neq X_1$
$(x_0,x_1) \leftarrow\!\!\$\ \mathbb{Z}_p^2$	**return** $\bot$	**return** $\bot$
$(X_0,X_1) := (x_0G, x_1G)$	**if** $X = X_0$	**if** $X = X_0$
$\beta' \leftarrow \mathsf{A}^{\mathrm{SIGN}_0,\mathrm{SIGN}_1,\mathrm{DH},\mathrm{DH}_{\mathrm{alg}},\mathrm{DH}^{\beta}_{\mathrm{chal}}}(\Gamma,(X_0,X_1))$	**return** $(x_0H = Y)$	**return** $(\bar{x}_0\boldsymbol{\eta} = \boldsymbol{\psi})$
return β'	**if** $X = X_1$	**if** $X = X_1$
	return $(x_1H = Y)$	**return** $(\bar{x}_1\boldsymbol{\eta} = \boldsymbol{\psi})$

Fig. 8. The game $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}}(\lambda)$ of Lemma 2. In the experiment, $\mathrm{DH}^0_{\mathrm{chal}} = \mathrm{DH}_{\mathrm{alg}}$ and $\mathrm{DH}^1_{\mathrm{chal}} = \mathrm{DH}$. The signing oracles $\mathrm{SIGN}_0, \mathrm{SIGN}_1$ are the same as in Figure 6.

in distinguishing the game $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}_{alg}}(\lambda)$ *is:*

$$\mathsf{Adv}^{\mathrm{algdh}}_{\Sigma,\mathsf{A}_{alg}}(\lambda) \leq q\left(3\mathsf{Adv}^{\mathrm{omdl}}_{\mathsf{GrGen}}(\lambda) + \frac{2}{p}\right) \quad .$$

We note that, in the game $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}_{\mathrm{alg}}}(\lambda)$, the oracle $\mathrm{DH}_{\mathrm{alg}}$ could be computed directly by the adversary itself. Its presence in the security experiment is merely didactic, and meant to illustrate the two possible instantiations of $\mathrm{DH}^{\beta}_{\mathrm{chal}}$.

The proof for this theorem can be found in Appendix A. This lemma will be used mostly to answer READ queries relying solely on the algebraic representation, and without knowing X_0, X_1 as they will themselves be used to embed a challenges in some security reductions. We are going to define the oracle SIMREAD as the oracle that behaves exactly as READ for Σ, but instead of checking the Diffie-Hellman over the group (i.e., using $\mathrm{DH}^0_{\mathrm{chal}}$) it does so over the algebraic representation (i.e., using $\mathrm{DH}^1_{\mathrm{chal}}$).

4.3 Privacy of the metadata bit

Theorem 1. *In the algebraic group model and the random oracle model, if DDH and OMDL are hard for* GrGen*, then the scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) has private metadata bit.*

Proof. The proof is done by means of a hybrid arguments, where via a sequence of intermediary steps we show that show $\mathrm{PMB}^{\beta}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda)$ is indistinguishable from $\mathrm{PMB}^{1-\beta}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda)$. To simplify the exposition, we consider the concatenation of CHAL with SIGN_2, assuming that the index i for the session is matched accordingly. Formally, the sequence of hybrids should check if SIGN_2 is invoked with a challenge session, and only if so run the code that is being modified.

Hyb_0 This is the game $\mathrm{PMB}^{\beta}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda)$, where CHAL is followed by SIGN_2. This is identified by the line between "•"'s, where we implicitly match sessions with the same index i. Additionally, we unroll the code of the subprocedures $\Sigma.\mathsf{Sign}_1$ and $\Sigma.\mathsf{Sign}_2$. Since the session i is added to S and later removed, we just ignored the two operations.

Hyb_1 We apply Lemma 2 and replace the oracle READ, which internally uses x_0, x_1, with the oracle SIMREAD, which instead recovers the private metadata bit solely relying on the algebraic representation of the tokens.

Hyb_2 In this hybrid, instead of honestly generating the commitment for $b = \beta$ as $[K_b; C_b] := k_b \cdot [G; H]$, we sample r^*_b, e^*_b from $\mathbb{Z}_p$ uniformly at random and set $[K_b; C_b] := r^*_b \cdot [G; H_b] + e^*_b \cdot [X_b; Y]$. Once we receive the challenge e (that, in turn, defines the challenge e_b), we equivocate $[K_b; C_b]$ by setting r_b to

$r_b^* + (e_b^* - e_b)x_b$. In fact, since $x_b[G; H_b] = [X_b; Y]$, we have that:

$$\begin{aligned}\begin{bmatrix} K_b \\ C_b \end{bmatrix} &= r_b^* \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b^* \begin{bmatrix} X_b \\ Y \end{bmatrix} \\ &= r_b \begin{bmatrix} G \\ H_b \end{bmatrix} - e_b^* x_b \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b x_b \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b^* \begin{bmatrix} X_b \\ Y \end{bmatrix} \\ &= r_b \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b \begin{bmatrix} X_b \\ Y \end{bmatrix}.\end{aligned} \tag{3}$$

(Nota bene: this is only for the case $b = \beta$.) The two distributions are perfectly indistinguishable: the commitment, as well as the response, are in fact distributed uniformly at random.

Hyb_3 In this hybrid, instead of defining H_0, H_1 as $\mathsf{H}_s(s)$, we sample H_0, H_1 uniformly at random from $\mathbb{G}$. We program the random oracle to answer with (H_0, H_1) when s is queried. We compute the rest just as before.

The only way for A to distinguishing the two games is to query H_s on s before it is programmed by Chal. If the total number of oracle queries of $\mathsf{A} \in \mathrm{PPT}$ is upper-bounded by $q \in \mathsf{poly}(\lambda)$, then,

$$\frac{q^2}{2^\lambda} \geq \left| \mathsf{Adv}^{\mathsf{Hyb}_3}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda) \right| ,$$

because s is sampled uniformly at random from $\{0,1\}^\lambda$.

Hyb_4 We now compute $H_{1-\beta}$ as $x_{1-\beta}^{-1} x_\beta H_\beta$ instead of sampling it uniformly at random. That is, $H_{1-\beta} \coloneqq x_{1-\beta}^{-1} Y$. (We stress that, for privacy of the metadata bit, the challenger selects the secret key, and in particular chooses both x_β and $x_{1-\beta}$.) We claim that the two hybrids are indistinguishable by DDH. We construct an adversary B against the DDH game. Let (P, A, B, C) be the challenge, which B randomizes as $\{(P, A, B_i', C_i')\}_i$ using the random self-reducibility of DDH. B samples $x_\beta \leftarrow \mathbb{Z}_p$ and sets $G \coloneqq P$, $X_{1-\beta} \coloneqq A$, $X_\beta \coloneqq x_\beta G$. For each query to Chal, B uses a fresh (P, A, B_i', C_i') and sets $H_\beta \coloneqq C_i'$, $Y \coloneqq x_\beta C_i'$ and $H_{1-\beta} \coloneqq x_\beta B_i'$. If (P, A, B_i', C_i') is a valid DDH sample, the distribution is the one of of Hyb_4. Otherwise, the distribution is that of Hyb_3. Therefore,

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

Hyb_5 At this point, we remark that $Y = x_0 H_0 = x_1 H_1$. In particular, now $x_{1-\beta}^{-1}[X_{1-\beta}; Y] = [G; H_{1-\beta}]$. We exploit this fact to equivocate $[K_{1-\beta}; C_{1-\beta}]$, similarly to what we already did in Hyb_2. This time, for $b = 1-\beta$, after sampling e_b^* and r_b^* uniformly at random, we set $r_b = r_b^* + (e_b^* - e_b)x_b$ and we remark that the verification equation is still satisfied. In fact, Equation (3) still holds (except that, here, $b = 1 - \beta$). The two hybrids are perfectly indistinguishable.

With Hyb_5, the security experiment is entirely independent of b. It follows that $\mathrm{PMB}^b_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda)$ and $\mathrm{PMB}^{1-b}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda)$ are indistinguishable with advantage:

$$\mathsf{Adv}^{\mathrm{pmb}}_{\Sigma[\mathsf{GrGen}],\mathsf{A}}(\lambda) \leq \frac{2q^2}{2^\lambda} + 2\mathsf{Adv}^{\mathrm{ddh}}_{\mathsf{GrGen},\mathsf{B}}(\lambda) + 2\mathsf{Adv}^{\mathrm{simread}}_{\Sigma,\mathsf{A}_{\mathrm{alg}}}(\lambda) ,$$

for any PPT adversary A that makes at most q oracle queries.

4.4 Unforgeability

Theorem 2. *In the algebraic group model and the random oracle model, if OMDL is hard for* GrGen *and if ROS is hard in dimension* ℓ *(where* ℓ *is the number of signing queries the adversary can do for each metadata bit), then the scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) is unforgeable.*

Oracle CHAL in Hyb_0

increment i
$s \leftarrow\$ \{0,1\}^\lambda$
$H_0, H_1 \coloneqq \mathsf{H}_s(s)$
$Y \coloneqq x_b \cdot H_b$
$k_b, r_{1-b}, e_{1-b} \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq k_b \begin{bmatrix} G \\ H_b \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq r_{1-b} \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_b \coloneqq e - e_{1-b}$
$r_b \coloneqq k_b - e_b x_b$
return (e_0, e_1, r_0, r_1)

Oracle CHAL in Hyb_1

$s \leftarrow\$ \{0,1\}^\lambda$
$H_0, H_1 \coloneqq \mathsf{H}_s(s)$
$Y \coloneqq x_b \cdot H_b$
$r_b^*, e_b^*, r_{1-b}, e_{1-b} \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq r_b^* \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b^* \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq r_{1-b} \cdot \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \cdot \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_b \coloneqq e - e_{1-b}$
// Equivocate r_b because $(G; H_b) \xmapsto{x_b} (X_b; Y)$
$r_b \coloneqq r_b^* + (e_b^* - e_b) x_b$
return (e_0, e_1, r_0, r_1)

Oracle CHAL in Hyb_2

$s \leftarrow\$ \{0,1\}^\lambda$
$H_0, H_1 \leftarrow\$ \mathbb{G}$; $Y \coloneqq x_b \cdot H_b$
Program $\mathsf{H}_s(s) \coloneqq (H_0, H_1)$
$r_b^*, e_b^*, r_{1-b}, e_{1-b} \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq r_b^* \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b^* \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq r_{1-b} \cdot \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \cdot \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_b \coloneqq e - e_{1-b}$
// Equivocate s_b because $(G; H_b) \xmapsto{x_b} (X_b; Y)$
$r_b \coloneqq r_b^* + (e_b^* - e_b) x_b$
return (e_0, e_1, r_0, r_1)

Oracle CHAL in Hyb_3

$s \leftarrow\$ \{0,1\}^\lambda$
$H_b \leftarrow\$ \mathbb{G}$; $Y \coloneqq x_b \cdot H_b$; $H_{1-b} \coloneqq x_{1-b}^{-1} Y$
Program $\mathsf{H}_s(s) \coloneqq (H_0, H_1)$
$r_b^*, e_b^*, r_{1-b}, e_{1-b} \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq r_b^* \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b^* \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq r_{1-b} \cdot \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b} \cdot \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_b \coloneqq e - e_{1-b}$
// Equivocate s_b because $(G; H_b) \xmapsto{x_b} (X_b; Y)$
$r_b \coloneqq r_b^* + (e_b^* - e_b) x_b$
return (e_0, e_1, r_0, r_1)

Oracle CHAL in Hyb_4

$s \leftarrow\$ \{0,1\}^\lambda$
$H_b \leftarrow\$ \mathbb{G}$; $Y \coloneqq x_b \cdot H_b$; $H_{1-b} \coloneqq x_{1-b}^{-1} Y$
Program $\mathsf{H}_s(s) \coloneqq (H_0, H_1)$
$r_b, e_b, r_{1-b}^*, e_{1-b}^* \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq r_b \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq r_{1-b}^* \cdot \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} + e_{1-b}^* \cdot \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_{1-b} \coloneqq e - e_b$
// Equivocate r_{1-b} because $(G; H_{1-b}) \xmapsto{x_{1-b}} (X_{1-b}; Y)$
$r_{1-b} \coloneqq r_{1-b}^* + (e_{1-b}^* - e_{1-b}) x_{1-b}$
return (e_0, e_1, r_0, r_1)

Oracle CHAL in Hyb_5

$s \leftarrow\$ \{0,1\}^\lambda$
$H_b \leftarrow\$ \mathbb{G}$; $Y \coloneqq x_b \cdot H_b$; $H_{1-b} \coloneqq x_{1-b}^{-1} Y$
Program $\mathsf{H}_s(s) \coloneqq (H_0, H_1)$
$r_b, e_b, k_{1-b} \leftarrow\$ \mathbb{Z}_p$
$\begin{bmatrix} K_b \\ C_b \end{bmatrix} \coloneqq r_b \cdot \begin{bmatrix} G \\ H_b \end{bmatrix} + e_b \cdot \begin{bmatrix} X_b \\ Y \end{bmatrix}$
$\begin{bmatrix} K_{1-b} \\ C_{1-b} \end{bmatrix} \coloneqq k_{1-b} \cdot \begin{bmatrix} G \\ H_{1-b} \end{bmatrix}$
• send $(s, Y, C_0, C_1, K_0, K_1)$; obtain e •
$e_{1-b} \coloneqq e - e_b$
$r_{1-b} \coloneqq k_{1-b} - e_{1-b} x_{1-b}$
return (e_0, e_1, r_0, r_1)

Fig. 9. Hybrids 1 to 5 for the proof private metadata security of Σ for Theorem 1.

We recall that ROS can be solved in polynomial time when the number of open sessions $\ell > \log p$ (see [BLL+21]). For smaller ℓ, there are subexponential attacks [Wag02, BLL+21], and although they can be compensated by choosing a larger prime, this would be resulting in a quite inefficient scheme except when restricting ℓ to be a very small constant.

The scheme Σ and this unforgeability proofs are however useful as a stepping stone to construct the clause version in Section 5 that rely on a currently non-broken assumption, the modified ROS assumption.

A formal proof is provided in Appendix B. We now highlight the main proof ideas. There are essentially three ways the adversary A can win:

(a) A outputs $\ell + 1$ tokens that are valid and extract all to the same bit $b \in \{0, 1\}$;
(b) A outputs a token (t_i, σ_i) that is valid but for which the bit cannot be read successfully, which can happen in two ways:
  (b_1) $\sigma_i = (H_0, H_1, Y, \pi)$ with $Y = x_0 H_0$ and $Y = x_1 H_1$.
  (b_2) $\sigma_i = (H_0, H_1, Y, \pi)$ with $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$.

Using Lemma 2, we can switch between checking relations such as $Y = x_b H_b$ using the discrete logarithm x_b of X_b (oracle $\textsc{Dh}(X_b, H_b, Y)$) and checking it using the algebraic representation without having to know x_b (oracle $\textsc{Dh}_{\text{alg}}(X_b, H_b, Y)$).

We rule out (b_1) by remarking that if $\textsc{Dh}_{\text{alg}}(X_b, H_b, Y)$ is true for both $b = 0$ and $b = 1$, then the representation of Y needs to contain the degree-2 monomial $\bar{x}_0 \bar{x}_1$ which is impossible as this monomial is never given to the adversary.[6]

We then note that π can be seen as the Fiat-Shamir of a Sigma protocol for the statement "$\textsc{Dh}(X_0, H_0, Y)$ OR $\textsc{Dh}(X_1, H_1, Y)$ is true."[7] Thus, we can rule out (b_2) by remarking that if $\textsc{Dh}(X_b, H_b, Y)$ is false for both $b = 0$ and $b = 1$, then π is not sound, which is not possible by adaptive soundness of Fiat-Shamir.

Finally, we rule out (a) using an argument similar to the one for unforgeability of blind Schnorr signatures in [FPS20]. Let us assume without loss of generality that the adversary outputs $\ell + 1$ valid tokens for $b = 0$. We embed OMDL challenges in X_0 and the commitments K_b for any $\textsc{Sign}_0(b)$ query. Then we use the $\ell + 1$ valid tokens of the adversary to either solve ROS or OMDL. The reduction to ROS is much more subtle than in [FPS20] because of the OR proof. Concretely, it requires some careful simulation of the oracle H_e from the ROS oracle $\mathsf{H}_{\mathsf{ROS}}$. This simulation basically uses the fact that a query $\mathsf{H}_e(Y, H_0, H_1, K_0, K_1, C_0, C_1, t)$ can correspond to a single split of e as $e = e_0 + e_1$. All other options would yield invalid tokens. Furthermore, this split can be computed efficiently given the correct algebraic representation.

4.5 Unlinkability

Theorem 3. *In the algebraic group model and the random oracle model, if DDH is hard for* GrGen*, then the scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) is unlinkable.*

Roughly speaking, the extractor Ext for this proof is the PPT machine that, upon receiving as input a token together with the algebraic representation of the public parameters $(X_{0\,[x_0]}, X_{1\,[x_1]})$ (which, we recall, will depend on the sole group element sent to the adversary, i.e. $G \in \mathbb{G}$), checks if two tokens share the same private metadata by testing for $Y = x_0 H_0$ and $Y = x_1 H_1$. In addition, using the recovered private keys, it is possible for the challenger to compute a token $(H_0, H_1, Y, (\mathbf{e}, \mathbf{r}))$ independently from any of the messages sent by the adversary. We direct the curious reader to Appendix C for a formal proof.

[6] At first glance it may look like such argument only works in the generic group model and should not work in the algebraic group model because of the appearance of being purely algebraic test (as opposed to directly using a computational assumption). But this is not the case: Lemma 2 allows to switch testing to algebraic representation. It does so using computational assumptions.

[7] For a formal definition of Sigma protocols and the soundness of their transform, we direct the reader to Cramer [Cra97] and Unruh [Unr17]. We omit a formal introduction here as they are only required for the security proof and they are a standard, widely used primitive in cryptography.

Remark 2. It is possible to prove unlinkability of Σ outside of the algebraic group model, with a small variation to the scheme proposed in Fig. 7. To achieve the above, the public parameters (X_0, X_1) of key generation algorithm $\Sigma.\mathsf{KeyGen}$ are followed by a *non-interactive zero-knowledge argument of knowledge*[8] for the following relation parametrized by the group description $(\mathbb{G}, p, G)$:

$$\mathsf{R}_{\mathrm{DL}} \coloneqq \left\{ \left((x_0, x_1), (X_0, X_1)\right) \in \mathbb{Z}_p^2 \times \mathbb{G}^2 : x_0 G = X_0 \ \wedge \ x_1 G = X_1 \right\}.$$

Roughly speaking the extractor Ext is the knowledge extractor of the underlying proof system that uses the trapdoor td in order to extract the proof. The rest of the proof remains unchanged.

5 Clause protocol

In this section we describe the clause version of the construction in Section 4. This protocol, illustrated Fig. 7 has the advantage of relying on the modified ROS assumption rather than the regular ROS assumption for unforgeability. The modified ROS assumption was introduced by Fuchsbauer et al. [FPS20] and is not susceptible to the attacks of Schnorr [Sch01a] and Benhamouda et al. [BLL+21].

Informally, this variant consists in computing twice the same commitment phase. (The elements H_0, H_1, $Y \in \mathbb{G}$ will be actually computed only once as an optimization.) The user, upon receiving two commitments, runs the user algorithm, once for each of them. The two resulting challenges are then sent to the server, who in turn samples uniformly at random one bit $d \leftarrow\!\!\$ \{0,1\}$ and computes the response only for the d-th transcript. Finally, it sends the d-th response along with the bit d itself, and the user proceeds with the unblinding algorithm for the d-th blinded commitment and challenge. The verification procedure is exactly as in Σ.

5.1 Privacy of the metadata bit

Theorem 4. *The scheme* $\Sigma[\mathsf{GrGen}]$ *has private metadata bit with advantage:*

$$\mathsf{Adv}^{\mathrm{pmb}}_{\Lambda,\mathsf{GrGen}}(\lambda) \leq \mathsf{Adv}^{\mathrm{pmb}}_{\Sigma,\mathsf{GrGen}}(\lambda) \,.$$

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

Hyb_0 This is the initial PMB game, defined in Figure 6.

Hyb_1 Instead of sampling d' in $\textsc{Sign}_1$, we sample it within Sign_1 and store it within the session state for later. This hybrid is perfectly indistinguishable from the previous one.

Hyb_2 We now change the way the commitments are produced. For d'-side of the clause, compute the commitment as before:

$$\begin{bmatrix} K_b^{(d')} \\ C_b^{(d')} \end{bmatrix} \coloneqq k_b \begin{bmatrix} G \\ H_b \end{bmatrix}$$
$$\begin{bmatrix} K_{1-b}^{(d')} \\ C_{1-b}^{(d')} \end{bmatrix} \coloneqq r_{1-b} \begin{bmatrix} G \\ H_{1-b} \end{bmatrix} - e_{1-b} \begin{bmatrix} X_{1-b} \\ Y \end{bmatrix}.$$

On the other hand, for the $(1-d')$-side, sample $\gamma_b, \delta_b, \eta_b \leftarrow\!\!\$ \mathbb{Z}_p$ for $b = 0, 1$ and compute:

$$\begin{bmatrix} K_0^{(1-d')} \\ C_0^{(1-d')} \end{bmatrix} \coloneqq \gamma_0 \begin{bmatrix} K_0^{(d')} \\ C_0^{(d')} \end{bmatrix} + \delta_0 \begin{bmatrix} X_0 \\ Y \end{bmatrix} + \eta_0 \begin{bmatrix} G \\ H_0 \end{bmatrix}$$
$$\begin{bmatrix} K_1^{(1-d')} \\ C_1^{(1-d')} \end{bmatrix} \coloneqq \gamma_1 \begin{bmatrix} K_1^{(d')} \\ C_1^{(d')} \end{bmatrix} + \delta_1 \begin{bmatrix} X_1 \\ Y \end{bmatrix} + \eta_1 \begin{bmatrix} G \\ H_1 \end{bmatrix} \tag{4}$$

[8] For a formal definition of a NIZK and arguments of knowledge, we direct the curious reader towards [KLOR20, Sec. 2.2]

This hybrid is perfectly indistinguishable from the previous one: the distribution of $K_0^{(d')}, C_0^{(d')}, K_1^{(d')} C_1^{(d')}$ are identical as nothing changed from the previous hybrid. and $K_0^{(1-d')}, C_0^{(1-d')}, K_1^{(1-d')} C_1^{(1-d')}$ still follows the uniform distribution over $\mathrm{Span}([G, H_0], [X_0, Y])$, respectively $\mathrm{Span}([G, H_1], [X_1, Y])$

Hyb_3 It is now possible to produce a signature using $\Sigma.\mathsf{Sign}_0$ as an external oracle: for any query in the game $\mathrm{PMB}_{\Lambda,\mathsf{A}}(\lambda)$ of the form $\textsc{Sign}_0(b)$, the challenger queries the external oracle $\Sigma.\mathsf{Sign}_0(b)$ and stores the returned value as the d'-side, then computes the $1-d'$-side of the clause as per Eq. (4). For any query of the form $\textsc{Sign}_1(i, (e_0, e_1))$, the challenger forwards the query to the oracle $\Sigma.\mathsf{Sign}_1(e_{d'})$ and returns the result together with the clause bit d'. This oracle is perfectly indistinguishable from the previous one.

Hyb_4 We now change $\textsc{Chal}$, and instead of forwarding a query to $\Sigma.\mathsf{Sign}_0(\beta)$, it forwards the query to $\Sigma.\textsc{Sign}_0(1-\beta)$. From Theorem 1, the security loss of this hybrid is $\mathsf{Adv}^{\mathrm{pmb}}_{\Sigma,\mathsf{A}}(\lambda)$.

Hyb_5-Hyb_8 essentially perform the reverse changes of Hyb_3-Hyb_0. The final hybrid Hyb_8 is $\mathrm{PMB}^{1-\beta}_{\mathsf{A},\Lambda}(\lambda)$. It follows that:

$$\mathsf{Adv}^{\mathrm{pmb}}_{\Lambda,\mathsf{GrGen}}(\lambda) \leq \mathsf{Adv}^{\mathrm{pmb}}_{\Sigma,\mathsf{GrGen}}(\lambda).$$

5.2 Unlinkability

Theorem 5. *The scheme Λ is unlinkable with advantage:*

$$\mathsf{Adv}^{\mathrm{unlink}}_{\Lambda,\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{unlink}}_{\Sigma,2\ell}(\lambda)\ .$$

Proof. Let Ext be the extractor of Theorem 3, that is, the machine that used the algebraic representation of the public parameters (X_0, X_1) to extract the secret keys $(x_0, x_1) \in \mathbb{Z}_p^2$ and checks, for every token of the form $(t, (H_0, H_1, Y, (\mathbf{e}, \mathbf{r})))$, if (G, X_b, H_b, Y) is a CDH tuple. This can always be done as the algebraic representation is always correct and is solely expressed in base G.

Let A be an adversary for the game $\mathrm{UNLINK}^{\beta}_{\Lambda,\mathsf{A},\mathsf{Ext},\ell}(\lambda)$, that upon receiving as input the public parameters crs, interacts with the oracles $\textsc{User}_0, \textsc{User}_1, \textsc{Get}, \textsc{Chal}$ and finally returns a guess β'.

We construct a PPT adversary B for the game $\mathrm{UNLINK}^{\beta}_{\Sigma,\mathsf{B},\mathsf{Ext},2\ell}(\lambda)$. The adversary B internally runs A, with the public parameters received as input. Upon receiving a query of the form $\textsc{User}_0(i, (s, Y, \mathbf{C}^{(0)}, \mathbf{K}^{(0)}, \mathbf{C}^{(1)}, \mathbf{K}^{(1)}))$, it defines $\mathsf{msg}_0 := (s, Y, \mathbf{C}^{(0)}, \mathbf{K}^{(0)})$ and $\mathsf{msg}_1 := (s, Y, \mathbf{C}^{(1)}, \mathbf{K}^{(1)})$, and invokes the user oracle twice, once with inputs $(2i, \mathsf{msg}_0)$ and once with input $(2i+1, \mathsf{msg}_1)$. Upon receiving a query of the form $\textsc{User}_1(i, (\mathbf{e}, \mathbf{r}, d'))$, B forwards the query to the user oracle with input $(2i+d', (\mathbf{e}, \mathbf{r}))$ and internally stores in a table T_d the pair (i, d'), to mark the clause associated to the i-th session. Queries to the oracle $\textsc{Get}(i)$ are responded fetching the element $d \in \{0, 1\}$ such that element (i, d) is in T_d. If no such element exists, then return $\perp$ (note that there cannot be more than one possible match). Similarly we handle queries to $\textsc{Chal}$. B returns whatever guess A makes upon returning. It follows that:

$$\mathsf{Adv}^{\mathrm{unlink}}_{\Lambda,\mathsf{A},\mathsf{Ext},\ell}(\lambda) \leq \mathsf{Adv}^{\mathrm{unlink}}_{\Sigma,\mathsf{B},\mathsf{Ext},2\ell}(\lambda).$$

5.3 Unforgeability

Theorem 6. *In the algebraic group model and the random oracle model, if OMDL is hard for GrGen and if modified ROS is hard in dimension ℓ, then the scheme $\Lambda[\mathsf{GrGen}]$ is unforgeable.*

Proof (Sketch). We start by noting that the same indistinguishability argument of Lemma 2 can be applied also for this scheme. However, in Λ, the $\textsc{Sign}_0, \textsc{Sign}_1$ oracles will produce different commitment, and as a consequence the partial evaluation provided by reduce will be different too (cf. Hyb_1 in Page 25). More formally, consider the coefficients associated to the commitments $K_b^{(0)}, C_b^{(0)}, K_b^{(1)}$, and $C_b^{(1)}$, for each private metadata bit $b \in \{0, 1\}$. The commitments $C_b^{(0)}$ and $C_b^{(1)}$ can be expressed in base $K_b^{(0)}$, resp. $K_b^{(1)}$ by computing $C^{(0)} = h_0 K^{(0)}$ and $C^{(1)} = h_1 K^{(1)}$, where the random oracle is programmed such that $h_0 G = H_0$ and $h_1 G = H_1$ for the respective session. The commitments $K_b^{(0)}, K_b^{(1)}$ can be replaced by their respective verification equation (whenever a session is complete), or expressed as two different unknowns in the algebraic

representation (whenever a session is not yet complete). Later in the proof, whenever Lemma 2 proceeds with a reduction to OMDL, one considers the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B},2\ell}(\lambda)$, where on each round the commitments $K_b^{(0)}, K_b^{(1)}$ produced by $\textsc{Sign}_0$ are generated both (for each clause) through $\textsc{Target}()$. Later, the clause selected by the signer is dealt as before, using the $\textsc{Help}()$ oracle so that the verification equation is satisfied. In the non-clause part, B obtains its discrete log base G through $\textsc{Help}$.

As in Section 4.4, there are three different attack strategies that satisfy the winning condition:

- A produces a token for which the bit cannot be read successfully, i.e. either $\sigma_i = (H_0, H_1, Y, \pi)$ is such that $Y = x_0 H_0$ and $Y = x_1 H_1$, or such that $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$.
- A outputs $\ell + 1$ tokens $(t_i, \sigma_i)_{i \in S}$ for some set $S \subset [m]$ such that the tokens are valid and all read for the same bit $b \in \{0, 1\}$.

The first case can be ruled out using a hybrid argument that follows exactly the same argument of Items Hyb_0 to Hyb_4 of Theorem 2. If the adversary produces a token such that $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$, then it is possible to construct an adversary for adaptive soundness for the non-interactive argument defined in Eq. (8). Secondly, the case $Y = x_0 H_0$ and $Y = x_1 H_1$ can be ruled out as impossible, as it leads to an incorrect degree in the algebraic representation (would imply a nonzero coefficient for the term $\bar{x}_0 \bar{x}_1$ but this is not possible as the algebraic representation is linear in $\bar{x}_0$ and $\bar{x}_1$).

In the second case, A outputs $\ell + 1$ valid tokens that read for the same bit. We construct an intermediate hybrid (similarly to Hyb_8 in Page 29) where the experiment aborts if two equations (displayed below, Eqs. (5) to (6)) are satisfied. Given the tokens (t_i, σ_i) indexed in $i \in S$ (S is the subset of $\ell + 1$ valid forgeries made by the adversary, for which the private metadata bit is the same), the challenger considers their algebraic representation, which depends on the public parameters and the commitments sent. Denote with $d_{j_b} \in \{0, 1\}$ the clause selected at the j_b-th issuance session, where $b = 0, 1$ and $j_b \in [\ell]$. In the i-th forgery (t_i, σ_i), given the associated commitments $K_b^{(i)}$ (for $b = 0, 1$), we denote with $\iota_{j_b,0}^{(i)\,(d_{j_b})}, \iota_{j_b,1}^{(i)\,(d_{j_b})}, \kappa_{j_b,0}^{(i)\,(d_{j_b})}, \kappa_{j_b,0}^{(i)\,(d_{j_b})}$ the coefficients of $K_0^{(j_b)\,(d_{j_b})}$, $K_1^{(j_b)\,(d_{j_b})}$, $C_0^{(j_b)\,(d_{j_b})}, C_1^{(j_b)\,(d_j)}$, and with $\varepsilon^{(i)}, \rho^{(i)}$ the coefficients of (respectively) X_0 and G. (Note that the algebraic representation for the commitments can always be obtained storing within the random oracle $\tilde{\mathsf{H}}_e$ the algebraic representation of the elements given. For a more explicit description, we direct the reader towards Page 29.) At the end of its execution, the adversary interacts ℓ times with the signing oracle. Denote with d_{j_b} the clause selected by the challenger during the session sess_{j_b} for the private metadata bit $b \in \{0, 1\}$ and $j_b \in [\ell]$. If, for each forgery indexed in $i \in S$:

$$e_0^{(i)} - \varepsilon^{(i)} - \sum_{j_0 \in [\ell]} \iota_{j_0,0}^{(i)\,(d_{j_0})} e_0^{(j_0)\,(d_{j_0})} + \kappa_{j_0,0}^{(i)\,(d_{j_0})} e_0^{(j_0)\,(d_{j_0})} h_0^{(j_0)} + \kappa_{j_0,0}^{(i)\,(d_{j_0})} e_1^{(j_0)\,(d_{j_0})} h_1^{(j_0)} + \sum_{j_1 \in [\ell]} \iota_{j_1,0}^{(i)\,(d_{j_1})} e_0^{(j_1)\,(d_{j_1})} = 0 \ , \tag{5}$$

$$\text{and for all } j_0 \in [\ell - 1] : \quad \iota_{j_0,0}^{(i)\,(1-d_{j_0})} = 0, \tag{6}$$

then the game aborts. Equation (5) is the analogues of Eq. (10) for the clause version, and it involves a linear combination of all the coefficients of X_0 when substituting the verification equations for the issued signatures with the algebraic representation of the forgeries (similarly to the argument of Page 30). We note that, differently from the proof of Theorem 2, here we have an additional superscript "d_{j_b}" that is referring to the binary clause selected by the signer at the j-th issuance session for the private metadata bit b. We also note that, as the clause $1 - d_{j_b}$ is never completed, the terms $\iota_{j_0,0}^{(i)\,(1-d_{j_0})}$ do not contribute in Equation (5) as they are assumed to be zero.

Equation (6), instead, asks that the adversary guesses correctly the clause used at the d_j-th session. If there exists a PPT adversary A that is able to distinguish the above hybrid, it is possible to construct a PPT adversary B for the game $\mathrm{mROS}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ defined in Fig. 2. The adversary B takes as input the group description $\Gamma = (\mathbb{G}, G, p)$, and generates the public parameters $X_0 = x_0 G, X_1 = x_1 G$ sampling at random $(x_0, x_1) \leftarrow_{\$} \mathbb{Z}_p$. Then, it internally runs the adversary $\mathsf{A}(\Gamma, (X_0, X_1))$. For any query to the signing oracles, it computes the responses honestly, following the protocol Λ. For any query to the random oracle H_s, it returns two random group elements, storing aside their discrete logarithm. Queries to the random oracle H_e are responded as described in Page 31. At the end of the game, the adversary A outputs, with

non-negligible probability, $\ell+1$ forgeries such that Equation (5) is satisfied. B outputs the mROS solution $(((\iota_{j_0,0}^{(i)\,(0)})_{j_0\in[\ell]}, (\iota_{j_0,0}^{(i)\,(1)})_{j_0\in[\ell]}, \chi^{(i)})_{i\in[\ell+1]}, (e^{(j_0)})_{j_0\in[\ell]})$. (For a detailed argument on why Eq. (5) leads to the correct solution, we direct the reader to Page 29: the reasoning is exactly the same, and considers only those commitments that belongs to the d_{j_0}-clause and for which a verification equation is provided at the end of signing query.)

At this point, it is possible to construct a reduction to $\mathrm{OMDL}_{\mathsf{GrGen},\ell}(\lambda)$. Consider B the adversary that internally runs the adversary $\mathsf{A}(\Gamma,(X_0,X_1))$ where $X_0 \leftarrow \textsc{Target}()$ and $X_1 \coloneqq x_1 G$ for some $x_1 \leftarrow_{\$} \mathbb{Z}_p$. Signature queries are handled using the $\textsc{Target}$ and $\textsc{Help}$ oracle (cf. Item Hyb_9 in Page 32) and read oracles are handled algebraically. At the end, B obtains a set $\ell+1$ valid forgeries, indexed by $i \in S$. For each i, the algebraic representation of the commitment $K_0^{(i)}$ is of the following form:

$$\begin{aligned} K_0^{(i)} \coloneqq \rho^{(i)} G + \varepsilon^{(i)} X_0 + \\ \sum_{\substack{b\in\{0,1\},d\in\{0,1\}\\ j_b\in[\ell]}} \iota_{j_b,0}^{(i)\,(d)} K_0^{(j_b)\,(d)} + \iota_{j_b,1}^{(i)\,(d)} K_1^{(j_b)\,(d)} + \\ \kappa_{j_b,0}^{(i)\,(d)} C_0^{(j_b)\,(d)} + \kappa_{j_b,1}^{(i)\,(d)} C_1^{(j_b)\,(d)}. \end{aligned} \tag{7}$$

Note that, in the above equation, we are indexing for each $j_b \in [\ell]$ (for $b = 0,1$), and for each clause $d \in \{0,1\}$, independently of whether it was selected in the response phase (that is, if $d = d_{j_b}$), or not (that is, if $d = 1 - d_{j_b}$). Expanding the algebraic representation with the verification equations (which holds by winning condition) we end up with two possible cases:

(i) Equation (5) does not hold, and thus there exists one i for which the algebraic representation leads to a non-trivial equation in X_0, G, thus allowing for the recovery of the discrete logarithm of X_0. In other words, it is possible to build an adversary for one-more discrete log.

(ii) Equation (5) holds but Equation (6) doesn't. Hence, by the additional condition introduced in the previous hybrid, there exists a i^*, j^* for which $\iota_{j_0^*,0}^{(i^*)\,(1-d_{j^*})}$ is nonzero. the adversary B for $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ runs $x_0 \coloneqq \textsc{Help}(X_0)$, thus obtaining the discrete log of X_0, and then $\textsc{Help}(K_{j_0,0}^{(1-d_j)})$ for all $i, j \neq i^*, j^*$. Using the above, B can recover the discrete log of $K_{j_0^*,0}^{(1-d_{j^*})}$ via Eq. (5), thus still breaking one-more discrete log.

Both cases can be simultaneously handled by the reduction. Hence, the adversary B wins the experiment $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$ every time A wins.

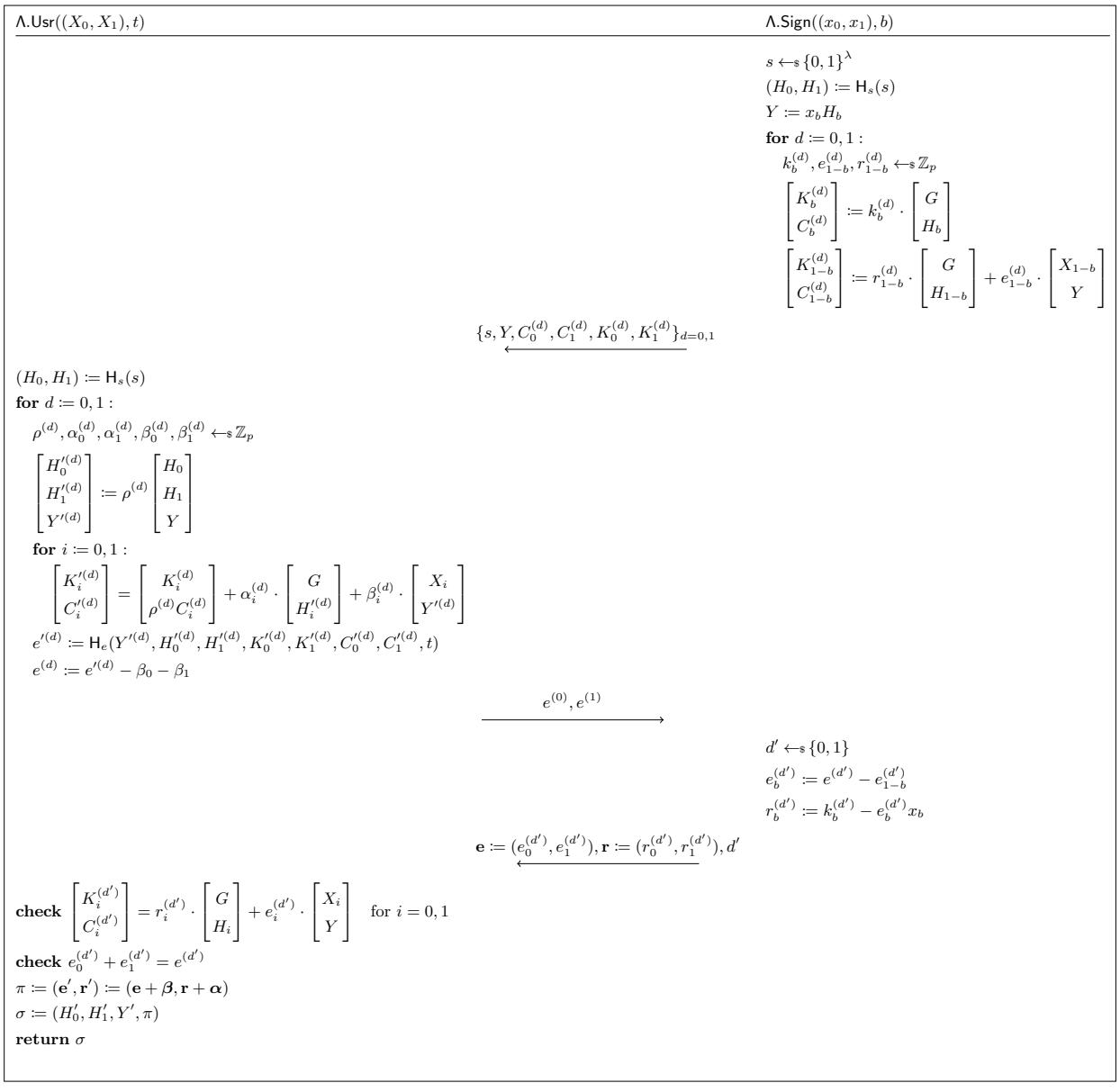

Fig. 10. The anonymous token protocol Λ (clause version).

References

ANN06. Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In *CT-RSA 2006*, 2006. 6

BL13. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013. 6

BLL+21. Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Heidelberg, October 2021. 2, 4, 17, 18

BNPS03. Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. 8

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003. 2

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Compact round-optimal partially-blind signatures. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 95–112. Springer, Heidelberg, September 2012. 2, 6

BV98. Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 59–71. Springer, Heidelberg, May / June 1998. 2, 8

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994. 3

Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. 1, 2, 6

Chr19. Google Chrome. Getting started with trust tokens. https://web.dev/trust-tokens/, 2019. 1

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001. 1

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993. 2, 6

Cra97. Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI Amsterdam, The Netherlands, 1997. 17, 29

DGS+18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018. 1, 3, 6, 10

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 8

FHS15. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015. 2

Fis06. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006. 2, 6

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. 2, 8

FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Heidelberg, May 2020. 2, 3, 4, 5, 6, 7, 17, 18, 25

Gro20. IETF Privacy Pass Working Group. Rfc base draft, issue 40. https://github.com/ietf-wg-privacypass/base-drafts/issues/40, 2020. 2

JKK14. Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253. Springer, Heidelberg, December 2014. 6

KLOR20. Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 308–336. Springer, Heidelberg, August 2020. 2, 3, 6, 10, 18

KW03. Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, 2003. 3

Oka93. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993. 2, 6

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000. 2, 6

PV05. Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2005. 2, 8

SC12. Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 133–150. Springer, Heidelberg, March 2012. 2, 6

Sch01a. Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, Heidelberg, November 2001. 2, 4, 6, 18

Sch01b. 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. 2, 6

Sch06. Claus Peter Schnorr. Enhancing the security of perfect blind dl-signatures. *Information Sciences*, 176(10):1305 – 1320, 2006. 2, 6

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. 6

Unr17. Dominique Unruh. Post-quantum security of Fiat-Shamir. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 65–95. Springer, Heidelberg, December 2017. 17, 29

Wag02. David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002. 2, 4, 17

A Proof of Lemma 2

Lemma 2. *Let the game* $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}}(\lambda)$ *be as defined in Fig. 8, where* $\mathrm{DH}^{0}_{\mathrm{chal}} = \mathrm{DH}$ *and* $\mathrm{DH}^{1}_{\mathrm{chal}} = \mathrm{DH}_{\mathrm{alg}}$. *For any algebraic adversary* A_{alg} *making at most* q *queries to the oracles* $\mathrm{DH}, \mathrm{DH}_{\mathrm{alg}}, \mathrm{DH}^{\beta}_{\mathrm{chal}}$, *the advantage in distinguishing the game* $\mathrm{AlgDH}^{\beta}_{\Sigma,\mathsf{A}_{alg}}(\lambda)$ *is:*

$$\mathsf{Adv}^{\mathrm{algdh}}_{\Sigma,\mathsf{A}_{alg}}(\lambda) \leq q\left(3\mathsf{Adv}^{\mathrm{omdl}}_{\mathsf{GrGen}}(\lambda) + \frac{2}{p}\right) \quad .$$

Proof. We provide a sequence of hybrid games that prove indistinguishability.

As we will be considering algebraic adversaries, with a random oracle H_e which takes as input group elements selected by the adversary $\mathsf{A}_{\mathrm{alg}}$ (that therefore must be explained as a linear combination of the elements given as input), we will assume that there exists another random oracle $\tilde{\mathsf{H}}_e$ that forwards the requests of the adversary to H_e omitting the linear combination (cf. Fuchsbauer et al. [FPS20, Theorem 1] for a formal argument).

Hyb_0 This is the game $\mathrm{AlgDH}^{0}_{\Sigma,\mathsf{A}}(\lambda)$.

Hyb_1 In this hybrid, we process queries H_s storing the discrete log for the elements given, and change the computation of C_b: instead of computing it as $C_b \coloneqq k_b H_b$, we use the discrete log of H_b. More precisely:
- In the random oracle H_s, sample $h_0, h_1 \leftarrow\!\!\$\, \mathbb{Z}_p$ and compute $H_0 \coloneqq h_0 G$, and $H_1 \coloneqq h_1 G$. Return $(H_0, H_1) \in \mathbb{G}^2$.
- C_b is computed as $C_b \coloneqq h_b K_b$ instead of $C_b \coloneqq k_b H_b$.

This hybrid is perfectly indistinguishable from the previous one.

With this hybrid, it is possible to express the discrete logarithm of any element sent by the challenger throughout the game as a linear combination of G, X_0, X_1 and the commitments K_b's (one per query) obtained during the signing query $\textsc{Sign}_0(b)$. In fact, all outputs of the random oracle H_s are computed sampling h_0, h_1 uniformly at random and returning $(h_0 G, h_1 G)$. The commitments for the bit $1-b$ are simulated: K_{1-b} is computed as $K_{1-b} = r_{1-b}G + e_{1-b}X_{1-b}$ and $C_{1-b} = r_{1-b}h_{1-b}G + e_{1-b}h_b X_b$. The commitment C_b is defined to be $C_b = h_b K_b$.[9]

More exactly, in any query $\mathrm{DH}_{\mathrm{alg}}(X_b, H_{[\boldsymbol{\eta}]}, Y_{[\boldsymbol{\psi}]})$ query we can consider $\boldsymbol{\eta}, \boldsymbol{\psi}$ as multilinear polynomials in G, X_0, X_1, and the commitments of the signing sessions not yet closed (that we denote for simplicity as $K^{(0)}, \ldots, K^{(m)}$, omitting the private metadata bit used for notational clarity). In other words $\boldsymbol{\eta}, \boldsymbol{\psi} \in \mathbb{F}[\bar{x}_0, \bar{x}_1, \bar{k}_0, \ldots, \bar{k}_m]$. To do so, we consider the partial evaluation, where we substitute all unknowns associated to the oracle replies of H_s with their respective discrete logarithms base G, and the commitments of currently closed sessions with their respective verification equations, e.g. a commitment K_b at the i-th completed session can be written as $r_b G + e_b X_b$, for the r_b, e_b associated to the i-th session. We denote this partial evaluation as a helper function reduce, that will be used by the challenger, and let $\hat{\boldsymbol{\eta}}_0 \coloneqq \mathsf{reduce}(\boldsymbol{\eta}_0)$; $\hat{\boldsymbol{\psi}} \coloneqq \mathsf{reduce}(\boldsymbol{\psi})$.

Let q be an upper-bound on the number of queries that $\mathsf{A}_{\mathrm{alg}}$ makes to the Diffie-Hellman oracles (more exactly, the $\mathrm{DH}, \mathrm{DH}_{\mathrm{alg}}, \mathrm{DH}^{\beta}_{\mathrm{chal}}$ oracles). We consider a sequence of $2q$ hybrids, where we incrementally switch (one by one) all Diffie-Hellman tests from testing on the group representation to the algebraic representation.

$[\mathsf{Hyb}_{3i+2}]_{i=0}^{q-1}$ We add one additional condition in the $(2i+1)$-th query to the Diffie-Hellman oracles (more exactly, one among $\mathrm{DH}, \mathrm{DH}_{\mathrm{alg}}$, or $\mathrm{DH}^{\beta}_{\mathrm{chal}}$): if it ever happens that a query is of the form $(X_0, H_{[\boldsymbol{\eta}]}, Y_{[\boldsymbol{\psi}]})$ such that $\hat{\boldsymbol{\psi}} = \psi_0 \bar{x}_0$ (for some $\psi_0 \in \mathbb{Z}_p$), and $\hat{\boldsymbol{\eta}} = \eta_1 \bar{x}_1 + \eta_0$ (for some $\eta_1 \in \mathbb{F}^*$, and $\eta_0 \in \mathbb{F}$), and $\psi_0 G = \eta_1 X_1 + \eta_0 G$, then the game immediately aborts and the adversary wins. Informally, with the above we are ruling out the particular case where $\boldsymbol{\eta}$ depends solely on $\bar{x}_1$.

[9] Looking ahead, when we will deal with the clause version the commitments will be doubled, and we will still have to study their algebraic representation as we do here. In that case, we are going to handle polynomials over X_0, X_1, and the commitments $K_b^{(0)}, K_b^{(1)}$, where b is the private metadata, for each $\textsc{Sign}_0$ query. Despite this minor change, the same argument applies.

If an adversary $\mathsf{A}_{\mathrm{alg}}$ can distinguish the above change from the previous hybrid, then it is possible to construct an adversary B for $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B},\ell}(\lambda)$. The adversary B receives as input the group description Γ and queries $\textsc{Target}$ for two challenges: $X_0 \coloneqq x_0 G$ ($x_0 \leftarrow\!\!\$\, \mathbb{Z}_p$), $X_1 \leftarrow \textsc{Target}()$. Then, it internally runs the adversary $\mathsf{A}_{\mathrm{alg}}(\Gamma, (X_0, X_1))$. If $X_0 = 0G$, then it aborts. (This happens with negligible probability $1/p$.) All queries of the form $\textsc{Sign}_0(1)$ are handled in the following way: B sets $K_1 \leftarrow \textsc{Target}()$, which is distributed uniformly at random (exactly as in Λ). The other commitments are computed as in the precious hybrid: $C_1 = h_1 K_1$ and K_0, C_0 are simulated as before. Queries of the form $\textsc{Sign}_0(0)$ are handled as before, using the secret key x_0. For any query of the form $\textsc{Sign}_1(j, e)$ where $e \in \mathbb{Z}_p$ is a challenge and j identifies the j-th session that is not yet closed. B defines $e_1 \coloneqq e - e_0$ and lets $r_1 \coloneqq \textsc{Help}(K_1 - e_1 X_1)$. Then, it returns $(\mathbf{e}, \mathbf{r})$. (Note that $\mathbf{e}$ is computed exactly as in the current hybrid and since $\textsc{Help}$ always output the correct discrete log, r_b follows the same distribution of the hybrid.)
During the execution of the $(2i+1)$-th $\textsc{Read}$ query, B stops the execution of $\mathsf{A}_{\mathrm{alg}}$ and attempts to compute a solution for the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$. (If the adversary returns an invalid query, or returns before the $(2i+1)$-th query, then B returns $\perp$.). If $\hat{\boldsymbol{\psi}} = \psi_0 \bar{x}_0$, $\hat{\boldsymbol{\eta}} = \eta_1 \bar{x}_1 + \eta_0$, and $\psi_0 G = \eta_1 X_1 + \eta_0 G$, then $\eta_1 \bar{x}_1 + \eta_0 = \psi_0$ and thus it is possible to recover the discrete logarithm of X_1 as $\bar{x}_1 \coloneqq (\psi_0 - \eta_0)/\eta_1$. Using x_1, B can compute the discrete log of all commitments (using the verification equation) and return the forgery $(\bar{x}_1, \bar{k}_1, \ldots, \bar{k}_\ell)$.

$[\mathsf{Hyb}_{3i+3}]_{i=0}^{q-1}$ We replace the $(2i+2)$-th equation of the Diffie-Hellman oracle (more exactly, of the $\textsc{Dh}$, $\textsc{Dh}_{\mathrm{alg}}$, or $\textsc{Dh}^{\beta}_{\mathrm{chal}}$ oracles): instead to checking that the input (X_0, H, Y) satisfies $Y = x_0 H$ in the group, we perform the (stronger) check that the equation also hold for the algebraic representations, i.e., that: $\hat{\boldsymbol{\eta}} \bar{x}_0 = \hat{\boldsymbol{\psi}}$.
Let $\mathsf{A}_{\mathrm{alg}}$ be an algebraic adversary for which this hybrid is noticeably different from the previous one. In other words, during its execution, with non-negligible probability the adversary $\mathsf{A}_{\mathrm{alg}}$ sends a query of the form $(H_{[\boldsymbol{\eta}]}, Y_{[\boldsymbol{\psi}]}, \pi))$ for which π is valid and $H_0 x_0 = Y$ but $\hat{\boldsymbol{\eta}} \bar{x}_0 \neq \hat{\boldsymbol{\psi}}$. We show that $\mathsf{A}_{\mathrm{alg}}$ can be used to construct a PPT adversary B for the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$. Let B be the adversary that, upon receiving as input the group description Γ, sets $X_0 \coloneqq \textsc{Target}()$ and $X_1 \coloneqq x_1 G$ (for $x_1 \leftarrow\!\!\$\, \mathbb{Z}_p$). (Note that the distribution of X_0 and X_1 is identical to the one of the hybrids $[\mathsf{Hyb}_{3i+3}]_{i=0}^{q-1}$.) Then, B internally runs the adversary $\mathsf{A}_{\mathrm{alg}}(\Gamma, X_0, X_1)$. B replies to random oracle queries in the same way as the challenger of the hybrid Hyb_1: for any query to H_s, B stores in a table the discrete logarithm of the group elements (H_0, H_1) and sends them to the adversary. Queries to the other oracles are replied in the following way (until the i-th execution):

- any query of the form $\textsc{Sign}_0(0)$ is handled in the following way: B sets $K_0 \coloneqq \textsc{Target}()$, which is still distributed uniformly at random. The other elements are computed in the same way of the hybrid Hyb_1: C_0 is computed as $C_0 = h_0 K_0$, and K_1, C_1 are part of the simulated proof:

$$\begin{bmatrix} K_1 \\ C_1 \end{bmatrix} = r_1 \begin{bmatrix} G \\ H_1 \end{bmatrix} + e_1 \begin{bmatrix} X_1 \\ Y \end{bmatrix},$$

  for $e_1, r_1 \leftarrow\!\!\$\, \mathbb{Z}_p^2$. ($K_b$ is distributed identically).
- For a query of the form $\textsc{Sign}_1(j, e)$, where $e \in \mathbb{Z}_p$ is a random challenge and j identifies the j-th session sess_j that is not yet closed, B checks if the bit b used in the first round and if $b = 1$ it proceeds as prescribed by the game Hyb_0. Otherwise, if $b = 0$, the query is handled with the help of the oracle $\textsc{Help}$ of the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$: B defines $e_b \coloneqq e - e_{1-b}$ and lets $r_b \coloneqq \textsc{Help}(K_b - e_b X_{1-b})$. Then, it returns $(\mathbf{e}, \mathbf{r})$. (Note that $\mathbf{e}$ is computed exactly as in the current hybrid and since $\textsc{Help}$ always output the correct discrete log, r_b follows the same distribution of the hybrid.)

If the adversary $\mathsf{A}_{\mathrm{alg}}$ halts before making the i-th query, then B fails returning $\perp$. If, during the i-th query, the verification equation is not valid, or if $\boldsymbol{\eta} \bar{x}_0 = \boldsymbol{\psi}$, then B fails returning $\perp$. Otherwise, during the execution of the i-th $\textsc{Read}$ query, B stops the execution of $\mathsf{A}_{\mathrm{alg}}$ and attempts to compute a solution for the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$. In other words, B has made a guess that $Y = \mathrm{CDH}(H_0, X_0)$ and $\hat{\boldsymbol{\eta}} \neq \boldsymbol{\psi} \bar{x}_0$. B completes all open sessions by sampling a random challenge $e_j \leftarrow\!\!\$\, \mathbb{Z}_p$ and querying $\textsc{Sign}_1(j, e_j)$, for any open session $\mathrm{sess}_j \neq \perp$. At this point, it is possible to write $\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\psi}}$ as univariate polynomials $\tilde{\boldsymbol{\eta}}, \tilde{\boldsymbol{\psi}} \in \mathbb{F}[\bar{x}_0]$

(that is, as a linear combination in basis G, X_0), performing another reduction step: B uses all verification equations from the commitments of sessions left open by the adversary before its execution was stopped and now closed, and the equation $X_1 = r_1 G$. We claim that $\tilde{\boldsymbol{\eta}}\bar{x}_0 - \tilde{\boldsymbol{\psi}}$ is a nontrivial polynomial that has at least one solution in the field $\mathbb{Z}_p$ that is also the discrete logarithm of X_0. To convince ourselves of the above, consider the following cases:

(i) if $\hat{\boldsymbol{\psi}} = \tilde{\boldsymbol{\psi}}$ and $\hat{\boldsymbol{\eta}} = \tilde{\boldsymbol{\eta}}$, then $\tilde{\boldsymbol{\psi}} \neq \tilde{\boldsymbol{\eta}}\bar{x}_0$.[10] Since $H_0 \neq 0$ then $\tilde{\boldsymbol{\eta}} \neq 0$ and thus $\tilde{\boldsymbol{\eta}}_0 x - \tilde{\boldsymbol{\psi}}$ is a polynomial of degree at least one and at most two. Since the representation of $\mathsf{A}_{\mathrm{alg}}$ is always valid, then whenever $\mathsf{A}_{\mathrm{alg}}$ wins the game then there must be a root of the above polynomial in the field that is the discrete log of X_0;

(ii) if $\hat{\boldsymbol{\psi}} \neq \tilde{\boldsymbol{\psi}}$ or $\hat{\boldsymbol{\eta}} \neq \tilde{\boldsymbol{\eta}}$, it must be the case that $\tilde{\boldsymbol{\psi}}$ or $\tilde{\boldsymbol{\eta}}$ has at least one non-zero coefficient of the commitments of a session that was not yet closed. (The case in which $\tilde{\boldsymbol{\psi}}$ or $\tilde{\boldsymbol{\eta}}$ has solely a non-zero coefficient for X_1 has been covered in the previous hybrid.) Thus, the representation of $\tilde{\boldsymbol{\psi}}, \tilde{\boldsymbol{\eta}}$ base (G, X_0) is given by coefficients selected uniformly at random *after* the execution of $\mathsf{A}_{\mathrm{alg}}$ has stopped. In fact, for all sessions sess_j, the challenges e_j are distributed uniformly at random and therefore the partial evaluation substituting $\bar{k}_j = r_j - e_j \bar{x}_0$ leads to a trivial polynomial $\tilde{\boldsymbol{\psi}} - \tilde{\boldsymbol{\eta}}_0 \bar{x}_0$ with negligible probability $1/p$. Otherwise, there must be exist a root in the field for $\tilde{\boldsymbol{\psi}} - \tilde{\boldsymbol{\eta}}\bar{x}_0$.

Wrapping up, whenever $\mathsf{A}_{\mathrm{alg}}$ wins at the i-th query, one of the roots $\tilde{\boldsymbol{\eta}}\bar{x}_0 - \tilde{\boldsymbol{\psi}}$ is (with overwhelming probability) the discrete logarithm of X_0. Thus B solves the polynomial in $\bar{x}_0$, checking that the solution is valid by checking $x_0 G = X_0$ and if it does, solves all equations $k^{(i)} = e^{(i)}_{b_i} x_{b_i} - r^{(i)}$ and returns the OMDL solution $(x_0, k^{(0)}, \ldots, k^{(\ell)})$.

$[\mathsf{Hyb}_{3\mathsf{i}+4}]_{i=0}^{q-1}$ Similarly to the previous hybrid, we now replace the check $Y = x_1 H_1$ with $\hat{\boldsymbol{\psi}} = \bar{x}_1 \hat{\boldsymbol{\eta}}$. Indistinguishability w.r.t. the previous hybrid essentially follows the same argument as before.
This time, however, the adversary B for the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ sets $X_0 \coloneqq \textsc{Target}()$ and $X_1 \coloneqq \textsc{Target}()$ and invokes the adversary $\mathsf{A}_{\mathrm{alg}}(\Gamma, (X_0, X_1))$. The queries to the $\textsc{Sign}_0(b)$, $\textsc{Sign}_1(j, e)$ and H_s are handled all in the same way. At the end of the execution, B closes all open signing sessions, samples $r_0 \leftarrow\!\!\$\, \mathbb{Z}_p$ and defines $r_1 \coloneqq \textsc{Help}(X_1 - r_0 X_0)$. Note that, differently from the previous hybrid, now X_1 is described base X_0, G. This time, B works on a different polynomial $\tilde{\boldsymbol{\eta}}(r_0 \bar{x}_0 + r_1) - \tilde{\boldsymbol{\psi}}$, which corresponds to the equation in the group $Y = \mathrm{CDH}(X_1, H_1)$. However, the underlying reasoning is the same: if the probability of distinguishing this check on the i-th query is non-negligible, then B can solve the polynomial and find a solution for the game $\mathrm{OMDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ with overwhelming probability.

The last hybrid is exactly the game $\mathrm{SIMREAD}^1_{\Sigma,\mathsf{A}}(\lambda)$. Therefore, the advantage of any algebraic adversary in distinguishing the game is:

$$\mathsf{Adv}^{\mathrm{simread}}_{\Sigma,\mathsf{A}_{\mathrm{alg}}}(\lambda) \leq q \left(3\mathsf{Adv}^{\mathrm{omdl}}_{\mathsf{GrGen},\mathsf{B}}(\lambda) + \frac{1}{p} \right),$$

where q is an upper-bound on the number of queries to the read oracle.

B Proof of Theorem 2

Theorem 2. *In the algebraic group model and the random oracle model, if OMDL is hard for* GrGen *and if ROS is hard in dimension* ℓ *(where* ℓ *is the number of signing queries the adversary can do for each metadata bit), then the scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) is unforgeable.*

Proof. We provide a proof in the algebraic group model by means of a hybrid argument:

[10] We recall again that $\hat{\boldsymbol{\psi}}$ accounts for the preliminary reduction: it is a multivariate polynomial where the unknowns are X_0, X_1 and the commitments of the sessions not yet closed. On the other hand, $\tilde{\boldsymbol{\psi}}$ is a univariate with a single unknown X_0.

Hyb_0 This is the initial unforgeability game $\mathrm{OMUF}_{\Sigma,\mathsf{A}}(\lambda)$ for the PPT adversary A. The adversary receives as input the public paramters $(X_0, X_1) \in \mathbb{G}^2$ and can interact at most ℓ times with the signing oracle ($\textsc{Sign}_0$ and $\textsc{Sign}_1$) for the same bit $b \in \{0,1\}$. Additionally, it has access to a $\textsc{Read}$ oracle that reads the bit hidden from the user in a token. At the end of its execution, A outputs m signatures (t_i, σ_i) at least one of the following holds:

(a) A outputs $\ell + 1$ tokens $(t_i, \sigma_i)_{i \in S}$ (with $S \subset [m]$) that are valid and extract all to the same bit $b \in \{0,1\}$;

(b) A outputs a token (t_i, σ_i) (with $i \in [m]$) that is valid but for which the bit cannot be read successfully. This can happen basically in two ways:

(b_1) $\sigma_i = (H_0, H_1, Y, \pi)$ with $Y = x_0 H_0$ and $Y = x_1 H_1$.

(b_2) $\sigma_i = (H_0, H_1, Y, \pi)$ with $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$.

In the first hybrid, we re-state the winning condition by writing out explicitly the conditions illustrated above.

Hyb_1 In this hybrid, we replace queries to $\textsc{Read}$ with their algebraic representation, using Lemma 2. More specifically, we consider the multivariate polynomials $\hat{\boldsymbol{\eta}}_0, \hat{\boldsymbol{\eta}}_1, \hat{\boldsymbol{\psi}} \in \mathbb{Z}_p[\bar{x}_0, \bar{x}_1, \bar{k}_1, \ldots, \bar{k}_m]$ associated to H_0, H_1 and Y, where the coefficients of $\bar{x}_0, \bar{x}_1$ are (respectively) the coefficients of X_0 and X_1 in the algebraic representation, and $\bar{k}_1, \ldots, \bar{k}_m$ are the coefficients of the sessions not yet closed. In this hybrid, instead of testing (in the group) $x_b H_b = Y$, for $b = 0, 1$, we test (in the algebraic representation) that $\hat{\boldsymbol{\eta}}_b \bar{x}_b = \hat{\boldsymbol{\psi}}$. We assume also that the challenger calls $\textsc{SimRead}$ on the returned tokens at the end of the game, to have a consistent behavior during the game execution and when checking the winning condition.

Hyb_2 We rule out (b_1) as impossible: we replace the winning condition that enforces bits are successfully read from valid signatures, and remove the condition (b_1), that essentially checks the forged tokens read simultaneously to both 0, and 1, that is, $\hat{\boldsymbol{\psi}} = \bar{x}_0 \hat{\boldsymbol{\eta}}_0$ and $\hat{\boldsymbol{\psi}} = \bar{x}_1 \hat{\boldsymbol{\eta}}_1$.

The Diffie-Hellman checks computed at the end of the game in order to read off the private metadata can be re-written more explicitly as:

(i) if $Y = \mathrm{DH}_{\mathrm{alg}}(X_0, H_0)$ and $Y = \mathrm{DH}_{\mathrm{alg}}(X_1, H_1)$, then we abort. This condition is exactly the one that in being covered in this current hybrid;

(ii) if $Y = \mathrm{DH}_{\mathrm{alg}}(X_0, H_0)$ and $Y \neq \mathrm{DH}_{\mathrm{alg}}(X_1, H_1)$ then we increase s_0 and account the token as reading off private metadata bit 0;

(iii) if $Y = \mathrm{DH}_{\mathrm{alg}}(X_1, H_1)$ and $Y = \mathrm{DH}_{\mathrm{alg}}(X_0, H_0)$ then we increase s_1;

(iv) if $Y \neq \mathrm{DH}_{\mathrm{alg}}(X_0, H_0)$ and $Y \neq \mathrm{DH}_{\mathrm{alg}}(X_1, H_1)$ then we do nothing.

If it exists an algebraic adversary $\mathsf{A}_{\mathrm{alg}}$ for which the winning probability in this hybrid is noticeably larger, then it during the i-th execution of the procedure $\Sigma.\mathsf{ReadBit}$, the challenger obtains a token such that $\sigma_i = (H_0, H_1, Y, \pi)$ and $\hat{\boldsymbol{\psi}} = \bar{x}_1 \hat{\boldsymbol{\eta}}_1 = \bar{x}_0 \hat{\boldsymbol{\eta}}_0$. Since the algebraic adversary $\mathsf{A}_{\mathrm{alg}}$ always provides a correct algebraic representation, if $\bar{x}_0 \mid \hat{\boldsymbol{\psi}}$ and $\bar{x}_1 \mid \hat{\boldsymbol{\psi}}$, then $\bar{x}_0 \bar{x}_1 \mid \hat{\boldsymbol{\psi}}$ (recall that $\hat{\boldsymbol{\eta}}_0 \neq 0$ and $\hat{\boldsymbol{\eta}}_1 \neq 0$). This is impossible, since the total degree of $\hat{\boldsymbol{\psi}}$ is one.

Hyb_3 We change Item (iv), and replace the check made using $\mathrm{DH}_{\mathrm{alg}}$ with a check using $\textsc{Dh}$. This follows from Lemma 2. Note that the lemma applies because we don't need the discrete logarithm of X_0 and X_1 at any point except in the calls of $\textsc{Sign}_0, \textsc{Sign}_1$, and $\textsc{Dh}$.

Hyb_4 In this hybrid, we rule out (b_2). If any of the tokens returned from A is such that $Y \neq \mathrm{CDH}(X_0, H_0)$, $Y \neq \mathrm{CDH}(X_1, H_1)$, then A can distinguish this hybrid from the previous one.

Consider A is a PPT adversary that takes as input $(X_0, X_1) \in \mathbb{G}^2$ and outputs a token $(t, (H_0, H_1, Y, \pi))$ with $Y \neq \mathrm{CDH}(X_0, H_0)$, $Y \neq \mathrm{CDH}(X_1, H_1)$. We note that, in this case, π can also be seen as a forgery for soundness of the sigma protocol for the following relation parametrized by the group description $(G, p, \mathbb{G})$:

$$\mathsf{R}_{\mathsf{or\text{-}dleq}} \coloneqq \{\big((b, x), (X_0, X_1, H_0, H_1, Y)\big) \in \{0,1\} \times \mathbb{Z}_p \times \mathbb{G}^5 : x[G; H_b] = [X_b; Y]\} . \tag{8}$$

In oder words, π can be seen as a proof for the language of all tuples $(X_0, X_1, H_1, H_1, Y) \in \mathbb{G}^5$ where $Y = \mathrm{CDH}(X_0, H_0)$ or $Y = \mathrm{CDH}(X_1, H_1)$. Since (in this particular hybrid) we consider the case where A outputs tokens not in the above language, then π constitutes a forgery of soundness, as $\Sigma.\mathsf{Ver}$ returns **true** by winning condition. We omit here the formal definition of an interactive sigma protocol (available

in Cramer [Cra97]) and the soundness of its Fiat-Shamir transform (proven in Unruh [Unr17, Thm. 21]), as they are standard tools required only for this step of the proof.
We construct the adversary B for the soundness game $\mathrm{SND}_{\Pi_{\text{or-dleq}},\mathsf{R}_{\text{or-dleq}},\mathsf{B}}(\lambda)$ in the following way: upon receiving as input a CRS, run $((X_0,X_1),(x_0,x_1)) \leftarrow \Sigma.\mathsf{KeyGen}(\mathsf{crs})$. Then, invoke the adversary A with the public parameters (X_0,X_1) and the CRS. Reply to the oracle queries $\textsc{Sign}_0$, $\textsc{Sign}_1$, $\textsc{Read}$ exactly as per Fig. 4, using the secret key honestly generated. At the end, the adversary A returns a token $(t_i,(H_0,H_1,Y,\pi))$ such that π verifies and $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$ (by Hyb_1, the probability that $\hat{\boldsymbol{\psi}} \neq \bar{x}_b \hat{\boldsymbol{\eta}}_b$ for $b=0,1$ and yet the corresponding equation in the group does not hold is negligible). B outputs a statement (X_0,X_1,H_0,H_1,Y) and a proof π.[11]

Hyb_5 We perform the reverse change of Hyb_3, and replace the check in Item (iv), made using $\textsc{Dh}_{\text{alg}}$ with a check using $\textsc{Dh}$. This follows from Lemma 2.

Hyb_6 At this point, the only winning condition for the adversary is (a). We impose an additional restriction: that the $\ell+1$ forgeries from the adversary A always read off the bit 0. This leads to a $1/2$ loss in advantage, and is equivalent to guessing the bit on which the forgeries are being given. To see this, consider a challenger that samples $b^* \leftarrow\!\!\$\, \{0,1\}$ and, if $b^*=1$, it swaps the public parameters X_0, X_1 and, in all oracle queries, instead of peforming them for the private metadata bit b it does so for the bit $1-b$. Hence, from now on, we are going to assume that the forgeries of the adversary are on $b=0$.

Hyb_7 We assume that for each forgery (t_i,σ_i), with $i \in [\ell+1]$, there exists an associated random oracle query of the form: $\mathsf{H}_e(\mathbf{K}^{(i)},\mathbf{C}^{(i)},\mathbf{H}^{(i)},Y^{(i)})$ (where the commitments can be computed from the verification equation). If such a query does not exist, then the challenger aborts the game and the adversary wins. The advantage of A is at most $(\ell+1)/p$, because it would imply that the adversary guessed correctly the output of the random oracle for one of the forged tokens provided by the adversary.
We additionally remark that, in algebraic group model, the oracle query does not only contain the group elements $(\mathbf{K}^{(i)},\mathbf{C}^{(i)},\mathbf{H}^{(i)},Y^{(i)})$ but also their respective algebraic representation, i.e. a linear combination of G, X_0, X_1, and the group elements sent during the ℓ issuance sessions.

Hyb_8 Now, if some equation described below (Eq. (10)) holds, the game aborts and the adversary wins immediately. For $i \in [\ell+1]$, denote where $\sigma_i = (H_0^{(i)},H_1^{(i)},Y^{(i)},\pi^{(i)})$ the i-th (valid) signature returned from the adversary, with $\pi^{(i)} = (\mathbf{e}^{(i)},\mathbf{r}^{(i)})$. By the previous hybrid, remark that, for each $i \in [\ell+1]$ it holds that $Y^{(i)} = \mathrm{CDH}(H_0^{(i)},X_0)$.
During the execution of the adversary, ℓ signatures are issued for each bit $b=\{0,1\}$. We index the *transcript* of the signing queries with private metadata bit $b=0$ in j_0, i.e.:

$$(\mathbf{K}^{(j_0)},\mathbf{C}^{(j_0)},\mathbf{H}^{(j_0)},\mathbf{e}^{(j_0)},\mathbf{r}^{(j_0)}) \qquad (j_0\text{-th transcript } \textsc{Sign}_0(0))$$

denotes (respectively) the group elements $((K_0^{(j_0)},K_1^{(j_0)}),(C_0^{(j_0)},C_1^{(j_0)}),(H_0^{(j_0)},H_1^{(j_0)}))$ issued during the j_0-th query to $\textsc{Sign}_0(0)$; $\mathbf{e}^{(j_0)},\mathbf{r}^{(j_0)}$ denotes the field elements provided during the respective query to $\textsc{Sign}_1$. Note that $K_0^{(j_0)}$ and $K_0^{(i)}$ are different elements even when $j_0=i$. The former is generated by the challenger as an answer to a $\mathsf{Sign}_0(0)$ query while the later is generated by the adversary. We abuse notation this way to reduce clutter of subscripts and superscripts.
Similarly, we index the transcript of the signing queries in $j_1 \in [\ell]$, i.e. the tuple:

$$(\mathbf{K}^{(j_1)},\mathbf{C}^{(j_1)},\mathbf{H}^{(j_1)},\mathbf{e}^{(j_1)},\mathbf{r}^{(j_1)}) \qquad (j_1\text{-th transcript } \textsc{Sign}_0(1))$$

denotes the group elements issued during the j-th query to $\textsc{Sign}_0(1)$ and the associated responses porovided during the relative query to $\textsc{Sign}_1$. By correctness of the signing algorithm, we have that for $b \in \{0,1\}$, $j_b \in [\ell]$:

$$\begin{bmatrix} K_0^{(j_b)} \\ C_0^{(j_b)} \end{bmatrix} = r_0^{(j_b)} \begin{bmatrix} G \\ H_0^{(j_b)} \end{bmatrix} + e_0^{(j_b)} \begin{bmatrix} X_b \\ Y^{(j_b)} \end{bmatrix}.$$

[11] Note to the expert reader: formally, the reduction must also take into account the input token $t \in \{0,1\}^\lambda$ that is submitted by A for any query to H_e. However, it is trivial to construct an oracle that samples $\tau \in \mathbb{Z}_p$ uniformly at random, stores the pair (t,τ) in a table for every request, and re-randomizes the challenge (and the forgery) accordingly.

Let $h_0^{(j_b)}G = H_0^{(j_b)}$ and $h_1^{(j_b)}G = H_1^{(j_b)}$ (which are selected by the challenger after Hyb_1) and $x_1 G = X_1$ (which is selected by the challenger at the beginning of the game). We note that the above equation can be formulated only base G and X_0, as:

$$\begin{bmatrix} K_0^{(j_b)} \\ C_0^{(j_b)} \end{bmatrix} = \begin{cases} \begin{bmatrix} r_0^{(j_0)}G + e_0^{(j_0)}X_0 \\ r_0^{(j_0)}h_0^{(j_0)}G + e_0^{(j_0)}h_0^{(j_0)}X_0^{(j_0)} \end{bmatrix} & \text{if } \ b = 0 \\ \begin{bmatrix} r_0^{(j_1)}G + e_0^{(j_1)}X_0 \\ (r_0^{(j_1)}h_0^{(j_1)} + e_0^{(j_1)}h_1^{(j_1)}x_1^{(j_1)})G \end{bmatrix} & \text{if } \ b = 1 \end{cases}$$

$$\begin{bmatrix} K_1^{(j_b)} \\ C_1^{(j_b)} \end{bmatrix} = \begin{cases} \begin{bmatrix} (r_1^{(j_0)} + e_1^{(j_0)}x_1)G \\ r_1^{(j_0)}h_1^{(j_0)}G + e_1^{(j_0)}h_0^{(j_0)}X_0^{(j_0)} \end{bmatrix} & \text{if } \ b = 0 \\ \begin{bmatrix} (r_1^{(j_1)} + e_1^{(j_1)}x_1)G \\ (r_1 h_1 + e_1^{(j_1)}h_1^{(j_1)}x_1)G \end{bmatrix} & \text{if } \ b = 1 \end{cases} \tag{9}$$

In the above, we deliberately expressed the equation solely in term of G, X_0, and replaced all other elements with their respective discrete logarithm, which is always known by the challenger as it is the one issuing sessions. Despite this choice might seem arbitrary, it will be useful for determining the ROS solution.

At the end of the execution, the adversary obtains the $\ell+1$ forgeries. The commitments of those forgeries always admit one (valid) algebraic representation in terms of the group elements of the $\textsc{Sign}_0$ queries, indexed in $j_b \in [\ell]$. We denote with $\iota_{j_b,0}^{(i)}, \iota_{j_b,1}^{(i)}, \kappa_{j_b,0}^{(i)}, \kappa_{j_b,1}^{(i)}$ the coefficients in the algebraic representation of respectively $K_0^{(j_b)}$, $K_1^{(j_b)}$, $C_0^{(j_b)}$, $C_1^{(j_b)}$, and with $\varepsilon^{(i)}, \rho^{(i)}$ the coefficients of (respectively) X_0 and G. It follows that:

$$K_0^{(i)} \coloneqq \rho^{(i)}G + \varepsilon^{(i)}X_0 + \sum_{\substack{b\in\{0,1\} \\ j_b\in[\ell]}} \iota_{j_b,0}^{(i)}K_0^{(j_b)} + \iota_{j_b,1}^{(i)}K_1^{(j_b)} + \kappa_{j_b,0}^{(i)}C_0^{(j_b)} + \kappa_{j_b,1}^{(i)}C_1^{(j_b)}.$$

In particular, using Equation (9), the coefficient of X_0 is:

$$\varepsilon + \sum_{j_0\in[\ell]} \underbrace{\iota_{j_0,0}^{(i)}e_0{}^{(j_0)}}_{K_0^{(j_0)}} + \underbrace{\kappa_{j_0,0}^{(i)}e_0{}^{(j_0)}h_0^{(j_0)}}_{C_0^{(j_0)}} + \underbrace{\kappa_{j_0,1}^{(i)}e_1{}^{(j_0)}h_1^{(j_0)}}_{C_1^{(j_0)}} + \sum_{j_1\in[\ell]} \underbrace{\iota_{j_1,0}^{(i)}e_0^{(j_1)}}_{K_0^{(j_1)}}$$

The above representation can be uniquely determined by the algebraic representation provided by A. The game aborts if, for all $i \in [\ell+1]$:

$$e_0^{(i)} - \varepsilon^{(i)} - \sum_{j_0\in[\ell]} \iota_{j_0,0}^{(i)}e_0{}^{(j_0)} + \kappa_{j_0,0}^{(i)}e_0{}^{(j_0)}h_0^{(j_0)} + \kappa_{j_0,1}^{(i)}e_1{}^{(j_0)}h_1^{(j_0)} + \sum_{j_1\in[\ell]} \iota_{j_1,0}^{(i)}e_0^{(j_1)} = 0\,. \tag{10}$$

Intuitively, this equation involves a linear combination of all the coefficients of X_0 in the algebraic representation of the commitment.

If the adversary A is able to distinguish this hybrid from the previous one, then it is possible to construct an adversary B that solves the game $\mathrm{ROS}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$ for the modulus $p = |\mathbb{G}|$. The adversary B takes as input the entire group description $\Gamma = (\mathbb{G}, G, p)$, identifying a cyclic group $\mathbb{G}$ of prime order p, and a generator $G \in \mathbb{G}$.

The adversary B generates the public parameters $X_0 = x_0 G$ and $X_1 = x_1 G$, storing aside their discrete logarithm. Then, it internally runs the adversary $\mathsf{A}(\Gamma, (X_0, X_1))$. For any query to the signing oracles $\textsc{Sign}_0$, $\textsc{Sign}_1$, it computes the commitments and the responses honestly, as described in $\Sigma.\mathsf{Sign}_0$

and $\Sigma.\mathsf{Sign}_1$. For any query to the random oracle H_s, it returns two group elements, while storing their respective discrete logarithm as done in Lemma 2. Queries to the random oracle H_e of the form $\mathsf{H}_e(Y, H_0, H_1, K_0, K_1, C_0, C_1, t)$ are responded in the following way. B checks if $Y \neq \mathrm{CDH}(H_1, X_1)$. This can be computed because B knows the discrete logarithm of X_1, and intuitively this case will capture all queries where $[K_1; C_1]$ are being simulated. If $Y = x_1 H_1$, the adversary returns a random element. Otherwise, if $Y \neq \mathrm{CDH}(H_1, X_1)$, define:

$$M = \begin{bmatrix} G & X_1 \\ H_1 & Y \end{bmatrix},$$

and consider the verification equation that would be associate to this query as a system of linear equations, i.e.

$$\begin{bmatrix} G & X_1 \\ H_1 & Y \end{bmatrix} \begin{pmatrix} r_1^* \\ e_1^* \end{pmatrix} = \begin{bmatrix} K_1 \\ C_1 \end{bmatrix} \tag{11}$$

Clearly (r_1^*, e_1^*) are not provided to the challenger, but they exist and are unique, since the matrix has full rank (G is not the identity element, and the columns are linearly independent as $Y \neq \mathrm{CDH}(H_1, X_1)$). Since B has access to the group representation of each of the elements in the matrix (as we are in the algebraic group model A will always provide valid algebraic representations in the group elements sent by B, of which B knows the discrete logarithm). Therefore, it is possible for B to invert the matrix associated to the discrete log of M, and compute the solution (r_1^*, e_1^*). Then, it replies to the random oracle query $\chi = (Y, H_0, H_1, K_0, K_1, C_0, C_1, t)$ with:

$$\mathsf{H}_e(\chi) = \mathsf{H}_{\mathsf{ROS}}((\iota_{j_0,0})_{j_0 \in [\ell]}, \chi) + \varepsilon + e_1^* + \Delta - \Theta \tag{12}$$

where:

$$\Theta \coloneqq \sum_{j_0 \in [\ell]} \iota_{j_0,0} e_1^{(j_0)}$$

$$\Delta \coloneqq \sum_{j_0 \in [\ell]} \kappa_{j_0,0}^{(i)} e_0^{(j_0)} h_0^{(j_0)} + \kappa_{j_0,1}^{(i)} e_1^{(j_0)} h_1^{(j_0)} + \sum_{j_1 \in [\ell]} \iota_{j_1,0}^{(i)} e_0^{(j_1)}$$

If the query is performed before the completion of the ℓ signing sessions, the coefficients of the algebraic representation are implicitly set to 0. Intuitively, Θ accounts for the simulated part in the i-th token, and Δ accounts for all terms which do not contribute into the ROS solution.
At the end of the game, the adversary outputs with non-negligible probability $\ell + 1$ tuples, such that Eq. (10) is satisfied. This implies that

$$\begin{aligned}
0 &= e_0^{(i)} - \varepsilon - \Big(\sum_{j_0 \in [\ell]} \iota_{j_0,0}^{(i)} e_0{}^{(j_0)} + \kappa_{j_0,0}^{(i)} e_0{}^{(j_0)} h_0^{(j_0)} + \kappa_{j_0,1}^{(i)} e_1{}^{(j_0)} h_1^{(j_0)} + \sum_{j_1 \in [\ell]} \iota_{j_1,0}^{(i)} e_0^{(j_1)} \Big) \\
&= (e^{(i)} - e_1^{(i)}) - \varepsilon - \left(\Delta + \sum_{j_0 \in [\ell]} \iota_{j_0,0}^{(i)} e_0^{(j_0)} \right) && \text{(capture known terms)} \\
&= e^{(i)} - e_1^{(i)} - \varepsilon - \left(\Delta - \Theta + \sum_{j_0 \in [\ell]} \iota_{j_0,0}^{(i)} e^{(j_0)} \right) && \text{(add and remove } \Theta\text{)} \\
&= \mathsf{H}_{\mathsf{ROS}}((\iota_{j_0,0}^{(i)})_{j_0 \in [\ell]}, \chi^{(i)}) - \sum_{j_0 \in [\ell]} \iota_{j_0,0}^{(i)} e^{(j_0)} && \text{(substitute Eq. (12))}
\end{aligned}$$

where $\chi^{(i)}$ is the input to the oracle H_e that matches the i-th token. Hence the tuple:

$$\left(((\iota_{j_0,0}^{(i)})_{j_0 \in [\ell]}, \chi^{(i)})_{i \in [\ell+1]}, (e^{(j_0)})_{j_0 \in [\ell]} \right)$$

constitutes a valid ROS solution, and B returns this tuple.

Hyb_9 Finally, we rule out (*a*) with a reduction to the OMDL problem. In other words, for any algebraic adversary A that outputs $\ell+1$ valid signatures, all reading private metadata bit $b=0$ is possible to construct an adversary B that wins OMDL every time A wins Hyb_8.

B receives as input the group description, and invokes $X \leftarrow \textsc{Target}()$. Then, it constructs the public parameters $(X_0 \coloneqq X, X_1 \coloneqq x_1 G)$ and invokes A with the same arguments. For any query to $\textsc{Sign}_0(0)$, B constructs $K_b \coloneqq \textsc{Target}()$ and $C_0 = h_0 K_0$, and K_1, C_1 are generated honestly as per $\Sigma.\mathsf{Sign}$. For any query to $\textsc{Sign}_1$, the adversary B computes the response $r_0 \coloneqq \textsc{Help}(K_0 - e_0 X)$, while e_1, r_1 are returned as per $\textsc{Sign}_1$. The rest of the protocol flows in the same way: H_0, H_1 are computed programming the random oracles responses in H_s, and storing their respective discrete log. Once A returns $\ell+1$ forgeries, B finds the first pair of indices i^* for which Eq. (10) is *not* satisfied, and recovers the discrete logarithm for X, $x^* \in \mathbb{Z}_p$. B considers the algebraic representation of the forged commitments $K_0^{(i)}$:

$$K_0^{(i)} \coloneqq \rho^{(i)} G + \varepsilon^{(i)} X_0 + \sum_{\substack{b\in\{0,1\} \\ j_b \in [\ell]}} \iota_{j_b,0}^{(i)} K_0^{(j_b)} + \iota_{j_b,1}^{(i)} K_1^{(j_b)} + \kappa_{j_b,0}^{(i)} C_0^{(j_b)} + \kappa_{j_b,1}^{(i)} C_1^{(j_b)}. \tag{13}$$

Because the signatures verify, $K_0^{(i)}$ must also satisfy the following equation:

$$K_0^{(i)} = r_0^{(i)} G + e_0^{(i)} X\ . \tag{14}$$

By substitution (plugging equations Eq. (9) and Eq. (14) into Eq. (13)), we obtain that the discrete log x^* of X. Finally, B recovers the discrete logarithm for all the commitments $K_{j_0,0}$ obtained via $\textsc{Target}$ by setting $k^*_{j_0,0} \coloneqq e_0^{(j_0)} x^* + r_0^{(j_0)}$ for each $j_0 \in [\ell]$, corresponding to the verification equation for the key X_0 used to sign the tokens. B returns the OMDL solution $(k^*_{b_0}, \ldots, k^*_{b_{\ell-1}}, x^*)$. Since the algebraic group representation provided by the adversary is always correct, then the solution is a valid OMDL solution.

At this point, we are left with a security experiment that always returns zero, independently from the adversary. It follows that the advantage of A in winning the $\mathrm{OMUF}_{\mathsf{A},\Sigma}(\lambda)$ is negligible.

C Proof of Theorem 3

Theorem 3. *In the algebraic group model and the random oracle model, if DDH is hard for* GrGen*, then the scheme* $\Sigma[\mathsf{GrGen}]$ *(described in Fig. 7) is unlinkable.*

Proof. Instead of proving that for any PPT adversary A it exits an extractor Ext that can recover the private metadata bit, we prove the stronger statement that there exists, in the algebraic group model, an extractor Ext for any PPT adversary A.

Let Ext be the extractor that, given as input the public parameters $(X_{0[x_0]}, X_{1[x_1]})$ together with their algebraic representation (that is, the discrete log x_0, x_1), and the token $(t, (H_{0[\boldsymbol{\eta}_0]}, H_{1[\boldsymbol{\eta}_1]}, Y_{[\boldsymbol{\psi}]}, \pi))$, it checks if $Y = x_0 H_0$ or $Y = x_1 H_1$ and returns the only bit for which it is satisfied (otherwise $\perp$). Since the algebraic representation is always correct, the extractor always recovers the correct bit, independently from the PPT adversary.

Additionally, the extractor Ext is independent from the adversary, which allows us to prove an even stronger formulation of unlinkability. Our proof proceeds by means of a hybrid argument:

Hyb_1 This is the original game $\mathrm{UNLINK}^0_{\Sigma,\mathsf{A},\ell}(\lambda)$. The adversarial issuer takes as input the CRS and interacts with the user multiple times. For any query to the oracle $\textsc{Chal}(i_0, i_1)$ made by the adversary, the challenger completes the session sess_{i_0} and sess_{i_1} and returns the respective tokens (in this order), provided that they are both for the same private metadata bit. At the end of its execution, it returns a guess β' and wins if $\beta' = 0$.

Hyb_2 In this game, we replace the random oracle H_e in the oracle $\textsc{User}_0$: we sample $e \leftarrow\!\!\$\, \mathbb{Z}_p$ uniformly at random and independently from $t \in \{0,1\}^\lambda$ and we program the random oracle H_e accordingly for the value e. We do so even if the value was already queried. The adversary has negligible advantage in distinguishing this hybrid from Hyb_1: for an adversary A that makes q_e queries to the oracle H_e and q queries to $\textsc{User}_0$ the advantage in distinguishing is:

$$\left|\mathsf{Adv}^{\mathsf{Hyb}_1}_{\Sigma,\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_2}_{\Sigma,\mathsf{A}}(\lambda)\right| \leq \prod_i^q \left(1 - \frac{i}{p}\right) + \frac{qq_e}{p}$$

Hyb_3 We now impose an additional condition on the game: if $Y = x_0 H_0 = x_1 H_1$, we immediately abort the game and return 1. Since H_0 and H_1 are distributed uniformly at random, this happens only if $H_1 = x_0 x_1^{-1} H_0$, that is with probability $1/p$. For an adversary making at most q queries to the random oracle H_s, the advantage in distinguishing this hybrid from Hyb_2 is:

$$\left|\mathsf{Adv}^{\mathsf{Hyb}_2}_{\Sigma,\mathsf{A}}(\lambda) - \mathsf{Adv}^{\mathsf{Hyb}_3}_{\Sigma,\mathsf{A}}(\lambda)\right| \leq \frac{q}{p}$$

Hyb_4 We impose yet another additional condition: in the oracle $\textsc{User}_1$, if $Y \neq x_0 H_0$ and $Y \neq x_1 H_1$, we immediately abort the game and return 1. If there exists an algebraic adversary A for which the output of this hybrid is noticeably different than the previous hybrid, then it is possible to construct an adversary B for soundness of the following sigma protocol:

$$\Pi_{\mathsf{or\text{-}dleq}} \coloneqq \mathrm{PoK}\{(b, x) : x_b[G; H_b] = [X_b; Y]\}$$

Let B be the adversary that internally invokes $\mathsf{A}_{\mathrm{alg}}$ on the crs received as input, and provides the same oracles of Hyb_3, and invokes the adversary $\mathsf{A}_{\mathrm{alg}}$ providing as input crs. During the (single) query to $\textsc{Init}$, B stores the public parameters, and in particular (X_0, X_1) with their algebraic representation (i.e. their discrete log x_0, x_1). If no query to $\textsc{Init}$ is made during the execution of $\mathsf{A}_{\mathrm{alg}}$, B returns $\perp$. If, during the execution of A, there exists a query to the oracle $\textsc{User}_1$ such that $\sigma = (H_0, H_1, Y, \pi)$ and $Y \neq x_0 H_0$, $Y \neq x_1 H_1$, then B returns the forged statement (X_0, X_1, H_0, H_1, Y) together with the forged proof π.[12]

Hyb_5 In this hybrid, we now compute directly Y, setting $Y \coloneqq x_b H'_b$, where $b \in \{0,1\}$ is the integer extracted by Ext. Such integer exists and it is unique: in fact, in the let $b \in \{0,1\}$ be the integer such that $Y = x_b H_b$. Such a b exists and is unique, because in Hyb_4 we removed the possibility that $x_0 H_0 \neq Y \neq x_1 H_1$ (hence it exists) and in Hyb_3 that $Y = x_0 H_0 = x_1 H_1$ (hence it is unique). This hybrid is perfectly indistinguishable from the previous one, since the extractor (and, in turn, the algebraic representation of $\mathsf{A}_{\mathrm{alg}}$) is always correct.

Hyb_6 We now sample H_0 and H_1 uniformly at random ourselves. This hybrid is indistinguishable by DDH.

Hyb_6 is entirely independent from the messages of the adversary, hence the protocol Σ is unlinkable.

[12] A similar argument was presented in Page 29.