Oblivious issuance of proofs

Michele Orrù[1], Stefano Tessaro[2], Greg Zaverucha[3], and Chenzhi Zhu[2]

[1] CNRS, Paris, France
[2] Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, USA
[3] Microsoft Research, Seattle, USA
michele@orru.net, tessaro@cs.washington.edu, gregz@microsoft.com, zhucz20@cs.washington.edu

Abstract. We consider the problem of creating, or *issuing*, zero-knowledge proofs *obliviously*. In this setting, a prover interacts with a verifier to produce a proof, known only to the verifier. The resulting proof is transferrable and can be verified non-interactively by anyone. Crucially, the actual proof cannot be linked back to the interaction that produced it. This notion generalizes common approaches to designing blind signatures, which can be seen as the special case of proving "knowledge of a signing key", and extends the seminal work of Camenisch and Stadler ('97). We propose a provably secure construction of oblivious proofs, focusing on discrete-logarithm representation equipped with AND-composition.

We also give three applications of our framework. First, we give a publicly verifiable version of the classical Diffie-Hellman based Oblivious PRF. This yields new constructions of blind signatures and publicly verifiable anonymous tokens. Second, we show how to "upgrade" keyed-verification anonymous credentials (Chase et al., CCS'14) to also be concurrently secure blind signatures on the same set of attributes. Crucially, our upgrade maintains the performance and functionality of the credential in the keyed-verification setting, we only change issuance. We observe that the existing issuer proof that the credential is well-formed may be verified by anyone; creating it with our framework makes it a blind signature, adding public verifiability to the credential system. Finally, we provide a variation of the U-Prove credential system that is provably one-more unforgeable with concurrent issuance sessions. This constitutes a fix for the attack illustrated by Benhamouda et al. (EUROCRYPT'21).

Beyond these example applications, as our results are quite general, we expect they may enable modular design of new primitives with concurrent security, a goal that has historically been challenging to achieve.

1 Introduction

Blind signatures, introduced by Chaum [17], are a fundamental tool in cryptography: they are a key component of e-voting applications, e-cash systems, anonymous credentials, and privacy-preserving protocols. Today, Google uses them to provide a VPN service that does not know of its users[4], Apple uses them as iCloud private relay[5], GNU for their e-cash system Taler[6] [22]. Blind signatures are currently undergoing standardization within IETF as Blind RSA signatures [21] and publicly-verifiable tokens [21]. With a surge of interest in privacy-preserving technologies, blind signatures have been extended to more involved use-cases: partially-blind signatures [2] tackle the case where part of the message is meant to be public; blind signatures with attributes (also known as Anonymous Credentials Light) [3] tackle issuance of signatures with partial attributes, and U-Prove [36], based on Brands credentials [10], provides a lightweight anonymous credential system. All these systems can be seen a proving more complex statements than "knowledge of the preimage of the verification key": the relation to be proven is more involved and often times the user also helps selecting the instance in a way that is oblivious to the issuer. We ask ourselves the following question:

Can proofs be issued obliviously, similarly to blind signatures?

[4] https://one.google.com/about/vpn
[5] https://developer.apple.com/news/?id=huqjyh7k
[6] https://taler.net/

The common denominator of many blind signature and anonymous credential systems is their resemblance to Σ-protocols, a family of zero-knowledge proofs [18] that is being executed between an *issuer* (acting as the prover) and a *user* (acting as a verifier). At the end, a non-interactively verifiable proof is created by the user. To achieve *oblivious issuance* of this final proof (i.e., to ensure that it cannot be linked back by the issuer to the interaction that generated it), the user carefully re-randomizes the proof transcript. Unfortunately, each of the above provides a different security analysis, which is often tedious and difficult. In some schemes security proofs are missing, and some others don't capture some realistic adversarial scenarios like interleaved open sessions (so-called *concurrent security*). A recent work of Benhamouda et al. [6] provided a concrete attack for some of these protocols.

Our contribution. In this work, we introduce the notion of oblivious proofs: these are proofs that are issued from a prover to the verifier without knowing the full statement. We show that this notion can be realised and provide a detailed security analysis for our framework.

Theorem 1 (informal). *There exists an oblivious proof scheme for algebraic relations.*

Algebraic relations are relations of the form $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)$ where $Z = \boldsymbol{Y} \cdot \boldsymbol{x}$ and $\boldsymbol{X} = M\boldsymbol{x}$, for some matrix $M \in \mathbb{G}^{n\times m}$ and vectors $\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}$. The proof size in linear in n. We provide three example applications for the above protocol:

- In Section 5 we design an efficient, publicly verifiable oblivious unpredictable function based on pairing-free curves, which is a publicly verifiable version of the classical Diffie-Hellman based Oblivious PRF. This yields, for example, a construction of a pairing-free blind signature scheme with a unique portion of the signature, as well as new constructions of anonymous tokens.
- In Section 6 we add *public verification* for CMZ [15] algebraic MACs, which are the basis for an efficient type of anonymous credential scheme called keyed-verification anonymous credentials. This type of credential system is used to privately manage group state in the Signal encrypted messaging application [16]. We provide a protocol for issuing CMZ credentials and a security analysis. Our protocol makes only modest changes to issuance, and does not require any change to existing keyed verification features (like credential presentation).
- In Section 7 we provide a variant of the U-Prove [36] issuance protocol from Microsoft Research that is secure in the concurrent setting. While the previous scheme was affected by a variant of the ROS attack [6], our variant comes with a proof of security in the concurrent setting.

As our results are quite general, we expect they may also be applicable to mend and thwart the attack of Benhamouda et al. [6], and also enable modular design of new schemes that are concurrently secure.

Technical Overview. We start from Σ-protocols for linear relations (e.g. as in Boneh–Shoup [9, Ch. 19]), i.e. relations of the for $M\boldsymbol{x} = \boldsymbol{X}$ where $\boldsymbol{x} \in \mathbb{Z}_p^m$ is the witness and $M \in \mathbb{G}^{n\times m}$ is a linear map. The transcript $(\boldsymbol{T}, e, \boldsymbol{r})$ satisfies the verification equation

$$M\boldsymbol{r} = \boldsymbol{T} + e\boldsymbol{X}.$$

Inspired by blind Schnorr [42] and Okamoto-Schnorr signatures [35] we blind the transcript with random $(\boldsymbol{\rho}, \varepsilon)$ such that

$$M(\boldsymbol{r} + \boldsymbol{\rho}) = (\boldsymbol{T} + M\boldsymbol{\rho} + \varepsilon\boldsymbol{X}) + (e - \varepsilon)\boldsymbol{X}$$

and thus obtain a proof $(\boldsymbol{T} + M\boldsymbol{\rho} + \varepsilon\boldsymbol{X}, e - \varepsilon, \boldsymbol{r} + \boldsymbol{\rho})$ that can be presented obliviously. In other words, users cannot be tracked in applications exposing these proofs. Two difficulties however arise here: (a) the instance $\boldsymbol{X}$ must be fully known to the prover, and this restricts the breath of possible applications, where often times also part of the instance must be re-randomized (e.g. for user-specific attributes in anonymous credentials, or public metadata in the signature); (b) one-more unforgeability of the resulting scheme is tricky. Concurrent security hinges on the ROS assumption [40,6] and thus inadequate for most practical instantiations.

To address the first issue, we consider relations where part of the instance (we call it the *argument*) is selected by the user. We consider relations of the form $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)$ where $Z = \boldsymbol{Y} \cdot X$ and $\boldsymbol{X} = M\boldsymbol{x}$. This allows us to address and extend previous constructions of blind signatures and anonymous credentials. To address the latter, we use the recent techniques of Tessaro and Zhu [43] for specifically Schnorr blind signatures. Roughly speaking, in the commitment phase, the prover commits also to an extra challenge $a \in \mathbb{Z}_p$, and engages in a proof for $M\boldsymbol{x}$ and $Z = \boldsymbol{Y} \cdot \boldsymbol{x}$ for some $\boldsymbol{Y}$ controlled by the user. Upon receiving a challenge e from the user, the server produces a response under challenge ae, which is uniformly distributed and unpredictable for the adversary. The resulting protocol is immune to ROS, but the proof of unforgeability demands a more tedious analysis for security (the proof is for more involved relations, and $\boldsymbol{Y}$ is now under control of the adversary), obliviousness (the element Z could be miscomputed), and correctness (the blinding of $\boldsymbol{X}$ must be multiplicative).

Related works. For more than two decades, it has been folklore in the cryptographic community that Σ-protocols may be issued obliviously, but this idea has not been investigated or ever formalized. Yet, it is a natural question to generalize Schnorr blind signatures in a similar way that Schnorr signatures were generalized to prove relations involving discrete logarithms. In fact, it was also not clear how security could be guaranteed for more than polylogarithmic concurrent queries due to an underlying assumption called ROS [38,42,6] that naturally emerges when studying concurrent security of interactive Σ-protocols.

Similar notions have emerged in the past literature. Belenkiy, Camenisch, and Chase [5] introduced the notion of *randomizable zero-knowledge proofs*, demanding that a zero-knowledge proof can be interactively re-randomized without knowing the witness. The notion applies naturally to Groth–Sahai proofs [27] and can be used to achieve delegatable anonymous credentials. To the best of our knowledge, no pairing-free randomizable proofs are known. De Santis and Yung [20] developed the concept of *meta proofs*, in which the holder of a proof can generate a proof that there exists a proof for the verification statement, however this is expensive (when it is possible at all), whereas our approach maintains nearly the same costs as the base Σ-protocol.

In the seminal work of Camenisch and Stadler [12] on proofs for statements about discrete logarithms, we see "blind issuance of Σ-protocols" is left as an open problem. However, proving concurrent soundness for these protocols has been historically hard due to the so-called ROS assumption, and a number of mitigations for it have been attempted over the years: Pointcheval [37] provided a variant of the Okamoto-Schnorr blind signature scheme [35] that boosts security using cut-and-choose to catch cheating behavior. Roughly speaking, in Pointcheval's fix, the *user* commits to two challenges at then beginning of the protocol and then has to open one of them. The resulting scheme, however requires the signer to stop issuing signatures to the user caught cheating, which is tricky in scenarios with a large, potentially anonymous, userbase. Abe [1] used OR-composition for Σ-protocols to yield an ordinary blind signature scheme under the DDH assumption that is secure in the concurrent setting, but with a tedious security analysis revisited in [31] and with small variations susceptible to attacks [3,6]. Katz, Loss, and Rosenberg [32] revisit the work of Pointcheval [37] and extend the cut-and-choose from 1-out-of-2 to 1-out-of-N, but require the server keep state of size N and increase it with the number of executions (and thus affecting the concrete communication complexity of the protocol). This line of work was refined further in recent works [14,28]. Fuchsbauer, Plouviez, and Seurin [24] proposed a framework for blind issuance of Schnorr signatures, relying on a different computational assumption called *modified ROS* (mROS). Roughly speaking, they have the *issuer* provide two possible commitments, and then give a response for only one of them, selected at random after the user has sent the challenge. However, the assumption requires larger parameters for concrete security, discouraging its use in practice. Fuchsbauer and Wolf [25] proposed a different approach, based on generic zero-knowledge proofs. Roughly speaking, the user proves that the challenge has been generated correctly while keeping private the blinding factors. This requires embedding the hash function inside the proof, which is expensive in practice and limits the provable security (since the hash function may not be formally modelled as a random oracle).

In our generic framework, we instead use Tessaro and Zhu's approach for mitigating the ROS attack [43]. Roughly speaking, here the issuer commits to a random value a when sending the commitment message of the Σ-protocol, and after receiving the challenge e from the user, sends a response that is valid under challenge ea. This effectively re-randomizes the challenges and thwarts ROS attacks making them statistically negligible [43]. We opt for this approach because, despite the fact that the final transcript is slightly different from the one of a Σ-protocol, it presents strong security guarantees while only requiring small changes to the initial Σ-protocol that are easily adoptable by protocol designers.

Table 1. Summary of variable names in this work. Variables marked with a prime ($'$) denoted the blinded values.

	Variable	Domain	Description
	$\Gamma = (\mathbb{G}, p, G)$		group description
	Λ		extra relation parameters
	$\boldsymbol{x}$	$\mathbb{Z}_p^m$	witness
	$\boldsymbol{X}$	$\mathbb{G}^n$	statement
	Z	$\mathbb{G}$	auxiliary statement
(υ)	$\boldsymbol{Y}$	$\mathbb{G}^{1\times m}$	(blinding factor of) auxiliary argument
	$[\boldsymbol{Y}; M]$	$\mathbb{G}^{(n+1)\times m}$	Σ-protocol morphism
	$\boldsymbol{T}$	$\mathbb{G}^{n+1}$	Σ-protocol commitment
(ε)	e	$\mathbb{Z}_p$	(blinding factor of) Σ-protocol challenge
$(\boldsymbol{\rho})$	$\boldsymbol{r}$	$\mathbb{Z}_p^m$	(blinding factor of) Σ-protocol response
(α, β)	$C = aH + bG$	$\mathbb{G}$	(blinding factors of) Pedersen commitment to a

2 Preliminaries

Notation. We denote by $(\mathbb{G}, p, G)$ the description of a group $\mathbb{G}$ of prime order p, with with generator G. We denote group operations additively, and given a scalar $x \in \mathbb{Z}_p$ we denote with xG scalar multiplication. We are going to use H to denote another group generator whose discrete logarithm base G is not known. To ease readability, we denote with 0 both the identity element in the group and in the field. We denote probabilistic algorithms in sans-serif, and by writing $y \leftarrow \mathsf{M}(x)$ we denote the act of sampling the value y from the probabilistic algorithm M on input x. The *range* of M on input x is denoted $[\mathsf{M}(x)]$.

The entries of a column vector $\boldsymbol{x}$ are denoted inline as $[x_1; x_2; x_3]$, entries of a row vector $\boldsymbol{x}^T$ as (x_1, x_2, x_3). We use $\boldsymbol{x}_{i..j}$ to denote the subvector of $\boldsymbol{x}$ consisting of elements from i to j. Assignment of a to the expression b is denoted as $a := b$; vectors are denoted in bold font. The identity matrix of size $n \times n$ over some field $\mathbb{Z}_p$ is denoted I_n (the field will be clear from the context). To ease readability, we denote with 0 the identity element in the group, in the field, and in the matrix space. It will be clear from the context which one is meant.

We assume that probabilistic algorithms run in time polynomial in the security parameter λ (abbrev p.p.t.) and have the security parameter implicitly as input. A summary of the variable names used throughout the protocol is available in Table 1.

The Decisional Diffie-Hellman problem. The Decisional Diffie-Hellman (DDH) problem is hard for a group generator GrGen if it is infeasible, given a group description $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ and a tuple $(aG, bG, C) \in \mathbb{G}^3$, to decide if $C = abG$ or C is a uniformly-random element of $\mathbb{G}$. More formally

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

where $\mathrm{DDH}^b_{\mathsf{oNIP},\mathsf{R},\mathsf{A}}(\lambda)$ is defined in Figure 2.This assumption is relevant only for one specific application of our framework described in Section 6.

The Discrete Logarithm problem. The Discrete Logarithm (DL) problem is hard for a group generator GrGen if it is infeasible, given a group description $\Gamma = (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ and a uniformly-random group element $X \leftarrow\!\!\$\; \mathbb{G}$, to compute $x \in \mathbb{Z}_p$ such that $xG = X$. More formally, for any p.p.t. adversary A

$$\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{DL}_{\mathsf{GrGen},\mathsf{A}}(\lambda) = \mathbf{true}\right] = \mathsf{negl}(\lambda) \ .$$

The q-Strong Discrete Logarithm problem. The q-Strong Discrete Logarithm problem (q-SDL) is hard for a group generator GrGen if it is infeasible, given a group description $\Gamma := (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ and the $q + 1$powers $G, xG, x^2G, \ldots, x^qG \in \mathbb{G}$, to compute $x \in \mathbb{Z}_p$ More formally, for any p.p.t. adversary A

$$\mathsf{Adv}^{\mathrm{sdl}}_{q,\mathsf{GrGen},\mathsf{A}}(\lambda) := \Pr\left[\mathrm{SDL}_{q,\mathsf{GrGen},\mathsf{A}}(\lambda) = \mathbf{true}\right] = \mathsf{negl}(\lambda) \ .$$

This assumption is solely used for $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ in Theorem 4.

Kernel Matrix Diffie-Hellman. The *Kernel Matrix Diffie-Hellman* (KMDH) problem [34, Def. 13] is hard for a group generator GrGen and a matrix distribution D if it is infeasible, given a group description $\Gamma \coloneqq (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ and a matrix $M \leftarrow \mathsf{D}(\Gamma)$ in $\mathbb{G}^{n\times m}$ to find non-trivial elements of the null space, that is, to exhibit an $\boldsymbol{r} \in \mathbb{Z}_p^n$ such that $M\boldsymbol{r} = 0$ and $\boldsymbol{r} \neq 0$. More formally, for any p.p.t. adversary A

$$\mathsf{Adv}^{\text{kmdh}}_{\mathsf{GrGen},\mathsf{D},\mathsf{A}}(\lambda) \coloneqq \Pr\big[\text{KMDH}_{\mathsf{GrGen},\mathsf{D},\mathsf{A}}(\lambda) = \textbf{true}\big] = \mathsf{negl}(\lambda) \enspace .$$

For the distributions we study, this assumption reduces to DL.

The ROS problem. The ROS (Random inhomogeneities in a Overdetermined Solvable system of linear equations) problem for dimension ℓ asks, given a prime number p and access to a random oracle H_{ros} with range in $\mathbb{Z}_p$, to find $(\ell + 1)$ vectors $\hat{\boldsymbol{\rho}}_j \in \mathbb{Z}_p^\ell$ for $j \in [\ell + 1]$, and a vector $\boldsymbol{e} = (e_1, \ldots, e_\ell)$ such that:

$$\mathrm{H}_{\text{ros}}(\hat{\boldsymbol{\rho}}_j) = \langle \hat{\boldsymbol{\rho}}_j, \boldsymbol{e}\rangle \qquad \text{for all } j \in [\ell + 1] \enspace .$$

While ℓ "trivial" solutions are easy to find by setting $\boldsymbol{\rho}_j \coloneqq (\delta_{1,j}, \delta_{2,j}, \ldots, \delta_{\ell,j})$ (where $\delta_{i,j}$ is the Kronecker delta) and $\boldsymbol{e} \coloneqq (\mathrm{H}(\boldsymbol{\rho}_1), \mathrm{H}(\boldsymbol{\rho}_2), \ldots, \mathrm{H}(\boldsymbol{\rho}_\ell))$, the hardness of the problem relies in finding a non-trivial linear combination of hash functions with range in $\mathbb{Z}_p$ for which is known a hash preimage. This problem was originally studied by Schnorr [42] in the context of blind signature schemes. Using a solver for the ROS problem, Wagner [45] showed that the unforgeability of the Schnorr and Okamoto–Schnorr blind signature schemes [41,35] can be attacked in subexponential time whenever more than $\mathsf{polylog}(\lambda)$ signatures are issued concurrently, using a generalization of the birthday paradox. More recently, Benhamouda et al. [6] provide a polynomial-time solver for the ROS problem. At the core of their attack, there is the observation that Wagner's attack fixes the vector $\boldsymbol{\rho}_{\ell+1} = (1, 1, 1, \cdots, 1)$ while the ROS problem offers much more flexibility in choosing arbitrary subsets and linear combinations of the elements in $\boldsymbol{c}$.

In this work, we study a variant of this problem that is *unconditionally hard for fields of large characteristic*, called weighted-fractional ROS [43, Section 3]. We display it in Figure 1, simplified for expositional purposes. We restate the main result here, for completeness.

Theorem 2 ([43, Theorem 1]). *For any $q > 0$ and prime number p, any p.p.t. adversary A for the game $\text{WFROS}_{q,p,\mathsf{A}}(\lambda)$ making at most q_h queries to the random oracle H, we have*

$$\mathsf{Adv}^{\text{wfros}}_{q,p,\mathsf{A}}(\lambda) \leq \frac{q_h(2q + q_h)}{p - 1} \enspace .$$

Roughly speaking, similarly to ROS the adversary is asked to provide a vector $\boldsymbol{e}$ and $\ell + 1$ linear combinations of its elements for which it is known a hash pre-image. However, here the linear combination is also re-randomized (*weighted*) by a random vector $\boldsymbol{s}$ chosen by the challenger whose elements are available to the adversary only once the corresponding element e_i has been set, which makes the problem unconditionally hard. Similarly to the ROS case, also here ℓ vectors are trivial to find: for $j = 1, \ldots, \ell$ set $\boldsymbol{a}_j \coloneqq \boldsymbol{b}_j \coloneqq (0, \delta_{1,j}, \delta_{2,j}, \ldots, \delta_{\ell,j})$, where $\delta_{i,j}$ is 1 for $i = j$ and 0 otherwise, and $e_j \coloneqq \mathrm{H}(\boldsymbol{a}_j, \boldsymbol{b}_j, j)$.

Σ-protocols. We briefly recap Σ-protocols, using the standard definition from Cramer [18] (as formalized by Boneh–Shoup [9]). Given a morphism described by a matrix $M \in \mathbb{G}^{n\times m}$, a Σ-protocol for the linear relation $\mathsf{R}_M = \{(\boldsymbol{x}, \boldsymbol{X}) \in \mathbb{Z}_p^n \times \mathbb{G}^n : M\boldsymbol{x} = \boldsymbol{X}\}$ is a 3-message protocol Σ between a prover and a verifier:

- $\Sigma.\mathsf{Prv}_0(\boldsymbol{x})$: the prover chooses a random vector $\boldsymbol{t} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and sends the *commitment* $\boldsymbol{T} \coloneqq M\boldsymbol{t}$ to the verifier.
- the verifier sends a random *challenge* $e \leftarrow\!\!\$\ \mathbb{Z}_p$
- $\Sigma.\mathsf{Prv}_1(st \coloneqq (\boldsymbol{x}, \boldsymbol{t}), e)$: computes and sends the *response* $\boldsymbol{r} \coloneqq \boldsymbol{t} + e\boldsymbol{x}$.

We call $\boldsymbol{T}$ the *commitment*, e the *challenge*, and $\boldsymbol{r}$ response. Together, we call the messages sent $(\boldsymbol{T}, e, \boldsymbol{r})$ the *transcript*. The transcript satisfies the verification equation:

$$M\boldsymbol{r} = \boldsymbol{T} + e\boldsymbol{X}.$$

Σ-protocols satisfy *2-special soundness* and *honest-verifier zero-knowledge* (cf. [18] for more information). The protocol can be compiled into a non-interactive zero-knowledge proof $(\mathsf{Prv}, \mathsf{Ver})$ via the Fiat-Shamir heuristic [23].

In the following, we will study how to issue such proofs obliviously. Since often times, in practical scenarios, parts of the statement will be decided at issuance time, we will slightly modify the above syntax to accommodate for more general relations.

```
Game WFROS_{q,p,A}(λ)
$\boldsymbol{s} \leftarrow\$ \mathbb{Z}_p^q$
$\boldsymbol{e} := (\bot)_{i\in[q]};\ Q_{\mathrm{H}} := [\,]$
$\{(\boldsymbol{a}_j, \boldsymbol{b}_j, aux_j)\}_{j\in[q+1]} \leftarrow \mathsf{A}^{\mathrm{H,S}}(p)$
return $\Big(\forall j \in [q]:\ e_j \neq \bot$ and
        $\forall j \neq k \in [q+1]:\ (\boldsymbol{a}_j, \boldsymbol{b}_j, aux_j) \neq (\boldsymbol{a}_k, \boldsymbol{b}_k, aux_k)$ and
        $\forall j \in [q+1]:\ \boldsymbol{b}_j \neq \boldsymbol{0}$ and
        $\forall j \in [q+1]:\ \langle \boldsymbol{a}_j, [1; \boldsymbol{s} \circ \boldsymbol{e}]\rangle = \mathrm{H}(\boldsymbol{a}_j, \boldsymbol{b}_j, aux_j) \cdot \langle \boldsymbol{b}_j, [1; \boldsymbol{s}]\rangle\Big)$

Oracle H(a, b, aux)
if $(\boldsymbol{a}, \boldsymbol{b})$ in $Q_{\mathrm{H}}$ :
    return $Q_{\mathrm{H}}[(\boldsymbol{a}, \boldsymbol{b}, aux)]$
$\delta \leftarrow\$ \mathbb{Z}_p$
$Q_{\mathrm{H}}[(\boldsymbol{a}, \boldsymbol{b}, aux)] := \delta$
return $\delta$

Oracle S(i, ε)
if $e_i \neq \bot$ or $i \notin [q]$ : return $\bot$
$e_i := \epsilon$
return $s_i$
```

Fig. 1. The WFROS$_{q,p,\mathsf{A}}(\lambda)$ game. The random oracle H has range in $\mathbb{Z}_p^*$.Vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^{q+1}$ are indexed from 0 to q, and $aux \in \{0,1\}^*$. Further, $\boldsymbol{e} \circ \boldsymbol{s} := [e_1 s_1; \ldots; e_q s_q]$.

```
Game SDL_{q,GrGen,A}(λ)
$(\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\$ \mathbb{Z}_p$
$x' \leftarrow \mathsf{A}(\mathbb{G}, p, G, xG, \ldots, x^q G)$
return $(x = x')$

Game DDH^b_{GrGen,A}(λ)
$(\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$a, b, c \leftarrow\$ \mathbb{Z}_p$
if $b = 0$ : return $(\mathsf{A}(\Gamma, (aG, bG, cG)) = 1)$
if $b = 1$ : return $(\mathsf{A}(\Gamma, (aG, bG, abG)) = 1)$
```

Fig. 2. Game q-SDL$_{\mathsf{GrGen},\mathsf{A}}(\lambda)$.

3 Oblivious issuance of proofs

Syntax. For a relation R whose elements are tuples of the form $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)$, denote

$$\begin{aligned}\mathsf{Core}(\mathsf{R}) &:= \{(\boldsymbol{x}, \boldsymbol{X})\ :\ \exists\, \boldsymbol{Y}, Z \text{ such that } (\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}\}\ , \text{and}\\ \mathsf{Arg}(\mathsf{R}) &:= \{\boldsymbol{Y}\ :\ \exists\, \boldsymbol{x}, \boldsymbol{X}, Z \text{ such that } (\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}\}\ .\end{aligned}$$

We call $\boldsymbol{Y}$ the *argument*, and Z the *augmented statement.* We say R is well-formed if for any $(\boldsymbol{x}, \boldsymbol{X}) \in \mathsf{Core}(\mathsf{R})$ and $\boldsymbol{Y} \in \mathsf{Arg}(\mathsf{R})$, there exists Z such that $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}$. As an example, the discrete logarithm equality (DLEQ) (employed in [13]) relation $\mathsf{R}_{\mathsf{dleq}}$ is indexed in the group description $\Gamma = (\mathbb{G}, p, G)$ and can be seen as a quadripartite relation whose elements $(x, X, Y, Z) \in \mathsf{R}_{\mathsf{dleq}}$ satisfy $x[G; Y] = [X; Z]$.

In this paper, we will actually deal with *families* of relations, i.e. relations R_{crs} parametrized by some common reference string $crs \in [\mathsf{oNIP.Setup}(1^\lambda)]$. For those, we assume the proof system is defined over $\mathsf{R} = \{\mathsf{R}_{crs}\}_{crs}$ and that the setup algorithm implicitly fixes the relation used during the protocol. For simplicity, when describing the relation we will sometimes omit the index crs.

Oblivious issuance of (non-interactive) proofs. An oblivious proof $\mathsf{oNIP} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Iss}, \mathsf{Usr}, \mathsf{Ver})$ for a relation R consists of:

- $crs \leftarrow \mathsf{oNIP.Setup}(1^\lambda)$, the setup algorithm, which generates the public parameters crs.
- $\pi \leftarrow \mathsf{oNIP.Prv}(crs, \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)$ the prover algorithm, that given as input a witness and an instance of R, produces a non-interactive proof π.
- two interactive p.p.t. algorithms $\mathsf{oNIP.Iss}$ (the *issuer*) and $\mathsf{oNIP.Usr}$ (the *user*) such that, given as input respectively witness and statement $(\boldsymbol{x}, \boldsymbol{X}) \in \mathsf{Core}(\mathsf{R})$, the user outputs an augmented statement Z for the argument $\boldsymbol{Y}$ chosen by the user together witha a non-interactive proof π. Since the argument $\boldsymbol{Y} \in \mathsf{Arg}(\mathsf{R})$ is chosen freely by the user, we call the oblivious issuance protocol *free* and we denote

```
Game $\mathrm{OBLV}^{b}_{\mathsf{oNIP,R,A}}(\lambda)$
$crs \leftarrow \mathsf{oNIP.Setup}(1^{\lambda})$
$b_0 := b;\ b_1 := 1 - b$
$b' \leftarrow \mathsf{A}^{\textsc{Init},\textsc{User}_0,\textsc{User}_1,\textsc{User}_2}(crs)$
$\mathbf{return}\ (b' = 1)$

Oracle $\textsc{Init}(\tilde{X}, \dbox{\tilde{Y}_0, \tilde{Y}_1}\ \boxed{\tilde{info}_0, \tilde{info}_1})$
$sess_0 := \mathtt{init};\ sess_1 := \mathtt{init}$
$\boldsymbol{X} := \tilde{\boldsymbol{X}}$
$\dbox{\boldsymbol{Y}_0 := \tilde{\boldsymbol{Y}}_0;\ \boldsymbol{Y}_1 := \tilde{\boldsymbol{Y}}_1}$
$\boxed{\text{Require: } \mathsf{Arg}(R, info_0) = \mathsf{Arg}(R, info_1)}$
$\boxed{info_0 := \tilde{info}_0;\ info_1 := \tilde{info}_1}$

Oracle $\textsc{User}_0(i)$
$\mathbf{if}\ i \notin \{0,1\}\ \mathbf{or}\ sess_i \neq \mathtt{init}:\ \mathbf{return}\ \bot$
$sess_i := \mathtt{sign1}$
$(st_i, Umsg) \leftarrow \mathsf{oNIP.Usr}_0(crs, \boldsymbol{X}, \dbox{\boldsymbol{Y}_{b_i}}, \boxed{info_{b_i}})$
$\mathbf{return}\ Umsg$

Oracle $\textsc{User}_1(i, Imsg_1)$
$\mathbf{if}\ i \notin \{0,1\}\ \mathbf{or}\ sess_i \neq \mathtt{sign1}:\ \mathbf{return}\ \bot$
$sess_i := \mathtt{sign2}$
$(st_i, Umsg_1) \leftarrow \mathsf{oNIP.Usr}_1(st_i, Imsg)$
$\mathbf{return}\ Umsg$

Oracle $\textsc{User}_2(i, Imsg_2)$
$\mathbf{if}\ i \notin \{0,1\}\ \mathbf{or}\ sess_i \neq \mathtt{sign2}:\ \mathbf{return}\ \bot$
$sess_i := \mathtt{closed}$
$((\hat{\boldsymbol{Y}}_{b_i}, Z_{b_i}), \pi_{b_i}) \leftarrow \mathsf{oNIP.Usr}_2(st_i, Imsg_2)$
$\mathbf{if}\ sess_0 = \mathtt{closed}\ \mathbf{and}\ sess_1 = \mathtt{closed}:$
  $\mathbf{if}\ ((\hat{\boldsymbol{Y}}_0, Z_0), \pi_0) = (\bot, \bot)\ \mathbf{or}\ ((\hat{\boldsymbol{Y}}_1, Z_1), \pi_1) = (\bot, \bot):$
    $\mathbf{return}\ (\bot, \bot)$
  $\dbox{\mathbf{if}\ \hat{\boldsymbol{Y}}_0 \neq \boldsymbol{Y}_0\ \mathbf{or}\ \hat{\boldsymbol{Y}}_1 \neq \boldsymbol{Y}_1: \mathbf{return}\ (\bot, \bot)}$
  $\mathbf{return}\ ((\hat{\boldsymbol{Y}}_0, Z_0, \pi_0), (\hat{\boldsymbol{Y}}_1, Z_1, \pi_1))$
$\mathbf{return}\ \mathtt{closed}$
```

Fig. 3. Game $\mathrm{OBLV}^{b}_{\mathsf{oNIP,R,A}}(\lambda)$. Free-mode ($Y$ is chosen by the user) contains everything but the $\boxed{\text{solid boxes}}$. Restricted-mode (the user can only choose a public information *info* related to Y) contains everything but the $\dbox{\text{dashed boxes}}$, where we addtionally $\mathsf{Arg}(R, info_0) = \mathsf{Arg}(R, info_1)$.

the interaction as:

$$((Z, \pi), \bot) \leftarrow \langle \mathsf{oNIP.Usr}(crs, \boldsymbol{X}, \boldsymbol{Y}), \mathsf{oNIP.Iss}(crs, \boldsymbol{x}, \boldsymbol{X}) \rangle$$

where $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}$. We also consider a more restricted setting, where $\boldsymbol{Y}$ is not chosen by the user freely but instead distributed uniformly in a set $\mathsf{Arg}(\mathsf{R}, info)$ determined by some public information *info*. We call the protocol *restricted* and denote the interaction as:

$$((Y, Z, \pi), \bot) \leftarrow \langle \mathsf{oNIP.Usr}(crs, \boldsymbol{X}, info), \mathsf{oNIP.Iss}(crs, \boldsymbol{x}, \boldsymbol{X}, info) \rangle$$

where $Y \in \mathsf{Arg}(\mathsf{R}, info)$ and $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}$.

– **true**/**false** $\leftarrow \mathsf{oNIP.Ver}(crs, \boldsymbol{X}, \boldsymbol{Y}, Z, \pi)$ outputs a bit to indicate whether π is a valid proof, that is, $\exists \boldsymbol{x}$ such that $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{R}$.

Security. Besides the standard notions of completeness and soundness for the tuple ($\mathsf{oNIP.Setup}$, $\mathsf{oNIP.Prv}$, $\mathsf{oNIP.Ver}$), we also ask that the issuance protocol is *correct*, that is: every honest execution of the issuance protocol leads to a verifying proof. Two notions are pivotal for security of oNIP: the issuer cannot link (Y, Z, π) to its respective issuance (*obliviousness*); the user does not gain sufficient knowledge of the witness to produce forgeries (*one-more unforgeability*). This can be seen as a generalization of blind signatures.

Below, we provide definitions for 2-round protocols both in free and restricted mode, indexing the i-message functions as $\mathsf{oNIP.Usr}_i$ and $\mathsf{oNIP.Iss}_i$: the user initiates the protocol via the procedure $\mathsf{oNIP.Usr}_0$ that takes as input $crs, \boldsymbol{X}$, and (respectively) $\boldsymbol{Y}$ and *info* in free and restricted mode. The procedure outputs some state $st_{u,0}$ together with some user message $Umsg_0$. The issuer, in turn, runs $\mathsf{oNIP.Iss}_1$ taking as input $crs, \boldsymbol{x}$ and the user message $Umsg_0$, returning $Imsg_1$ along with some state $st_{i,1}$. The protocol continues through $\mathsf{oNIP.Usr}_1(st_{u,0}, Imsg_1)$ and $\mathsf{oNIP.Iss}_2(st_{u,0}, Umsg_1)$, returning again a new state and the next message. Finally, the procedure $\mathsf{oNIP.Usr}_2(st_{u,1}, Imsg_2)$ returns either $\bot$ or a proof π.

```
Game $\mathrm{OMUF}_{\mathsf{oNIP,R,A}}(\lambda)$
$crs \leftarrow \mathsf{oNIP.Setup}(1^\lambda)$
$(\boldsymbol{x}, \boldsymbol{X}) \leftarrow\!\!\$\; \mathsf{Core}(\mathsf{R})$
$\ell := 0;\ \ Opn := \emptyset;\ \ Fin := \emptyset$
$\{(\boldsymbol{Y}_j^*, Z_j^*, \pi_j^*)\}_{j \in [\ell+1]} \leftarrow \mathsf{A}^{\textsc{Issuer}_1, \textsc{Issuer}_2}(crs, \boldsymbol{X})$
return $\Big(\forall\, j \in [\ell+1]\ :\ (\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}_j^*, Z_j^*) \in \mathsf{R}$, and
    // All relations are in the family, and ...
    $\forall\, j,k \in [\ell+1]\ j \neq k\ :\ (\boldsymbol{Y}_j^*, Z_j^*, \pi_j^*) \neq (\boldsymbol{Y}_k^*, Z_k^*, \pi_k^*)$, and
    // ... all proofs are different, and..
    $\forall\, j \in [\ell+1]\ :\ \mathsf{oNIP.Ver}(\boldsymbol{X}, \boldsymbol{Y}_j^*, Z_j^*, \pi_j^*) = \mathbf{true}\Big)$
    // ... all proofs verify.
```

```
Oracle $\textsc{Issuer}_1(i, Umsg_0, \boxed{info})$
if $i \in Opn \cup Fin$ :  return $\perp$
$Opn := Opn \cup \{i\}$
$(st_i, Imsg_1) \leftarrow \mathsf{oNIP.Iss}_1(crs, \boldsymbol{x}, Umsg_0, \boxed{info})$
return $Imsg_1$

Oracle $\textsc{Issuer}_2(i, Umsg_1)$
if $i \notin Opn$ :  return $\perp$
$\ell := \ell + 1;\ \ Opn := Opn \setminus \{i\}$
$Fin := Fin \cup \{i\}$
$Imsg_2 \leftarrow \mathsf{oNIP.Iss}_2(st_i, Umsg_1)$
return $Imsg_2$
```

Fig. 4. Game $\mathrm{OMUF}_{\mathsf{oNIP,R,A}}(\lambda)$. Free-mode ($Y$ is chosen by the user) contains everything but the solid boxes. Restricted-mode (the user can only choose a public information *info*) contains everything.

Obliviousness. Obliviousness means that proofs cannot be be linked back to the issuance session that created them even given the associated arguments. More specifically, a non-interactive proof oNIP with oblivious issuance for a quadripartite relation R is *oblivious* if for all p.p.t. adversaries A

$$\mathsf{Adv}^{\mathrm{oblv}}_{\mathsf{oNIP,R,A}}(\lambda) := \left|\Pr\left[\mathrm{OBLV}^0_{\mathsf{oNIP,R,A}}(\lambda) = \mathbf{true}\right] - \Pr\left[\mathrm{OBLV}^1_{\mathsf{oNIP,R,A}}(\lambda) = \mathbf{true}\right]\right| = \mathsf{negl}(\lambda)$$

where $\mathrm{OBLV}^b_{\mathsf{oNIP,R,A}}(\lambda)$ is defined in Figure 3.

Unforgeability. In one-more unforgeability, we demand that no p.p.t. adversary A can produce $\ell + 1$ different valid proofs after seeing ℓ interactions. We denote the advantage of A in winning the game $\mathrm{OMUF}_{\mathsf{oNIP,R,A}}(\lambda)$ by $\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oNIP,R,A}}(\lambda)$. That is:

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oNIP,R,A}}(\lambda) := \Pr\left[\mathrm{OMUF}_{\mathsf{oNIP,R,A}}(\lambda) = \mathbf{true}\right] = \mathsf{negl}(\lambda)$$

where $\mathrm{OMUF}_{\mathsf{oNIP,R,A}}(\lambda)$ is defined in Figure 4.

4 Oblivious issuance of proofs for algebraic relations

In this section, we will first introduce the notion of algebraic relations, then describe our oblivious issuance of proofs for the algebraic relations with two modes, and finally analyse the security of our scheme.

4.1 Algebraic relations

For any integers $n, m \geq 1$, a family of algebraic relations is a family of quadripartite relations, denoted as $\{\mathsf{AlgR}_{\Gamma,M}\}_{(\Gamma,M)}$, where $\Gamma = (\mathbb{G}, p, G)$ is a group description, M is a matrix in $\mathbb{G}^{n\times m}$, and

$$\mathsf{AlgR}_{\Gamma,M} := \left\{(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)\ :\ \boldsymbol{x} \in \mathbb{Z}_p^m\ ,\quad \boldsymbol{Y} \in \mathbb{G}^{1\times m}\ ,\quad \begin{bmatrix} Z \\ \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y} \\ M \end{bmatrix} \cdot \boldsymbol{x} \right\}\ .$$

($\boldsymbol{Y}$ is considered as a row vector.) In order to make the relation non-trivial, we require $M \neq 0G$, where $0 \in \mathbb{Z}_p^{n\times m}$ is the zero matrix.

Example 1. As an example, consider the Schnorr [39] and Okamoto-Schnorr [35] signature schemes. Both can be seen as "proofs" that some signing key associated to some verification key. In particular, these relations are algebraic: for blind Schnorr [39] the algebraic relation proven is indeed:

$$\mathsf{R}_{\mathsf{sch}} := \{(x, X, \perp, \perp)\ :\ x \in \mathbb{Z}_p\ ,\quad xG = X\}\quad ,$$

where $n = m = 1$ and $M = \begin{bmatrix} G \end{bmatrix}$. For Okamoto-Schnorr [35] the algebraic relation would be:

$$\mathsf{R}_{\mathrm{os}} := \left\{(\boldsymbol{x}, X, \perp, \perp) \;:\; \boldsymbol{x} \in \mathbb{Z}_p^2\,, \quad X = [G; W] \cdot \boldsymbol{x}\right\}\ ,$$

where $n = 1$, $m = 2$, $M = [G; W]$, and W is another generator of $\mathbb{G}$ for which the discrete-logarithm base G is not known.

Example 2. As yet another example, consider discrete logarithm equality (DLEQ) proofs used in VOPRF constructions [13,30]. We can define the DLEQ relations in the framework of the algebraic relations, where $n = m = 1$, $M = \begin{bmatrix} G \end{bmatrix}$, and the relation is

$$\mathsf{R}_{\mathrm{dleq}} := \left\{(x, X, Y, Z) \;:\; \forall\ x \in \mathbb{Z}_p \text{ and } Y \in \mathbb{G},\ \begin{bmatrix} Z \\ X \end{bmatrix} = \begin{bmatrix} Y \\ G \end{bmatrix} \cdot x\right\}\ .$$

4.2 Oblivious issuance protocol of proofs

Despite we are unable to show that general classical Σ-protocols described in Section 2 satisfy oblivious issuance, we can show that this is possible to do so for a small variation of them. We define a new oblivious issuance protocol of proofs for algebraic relations, denoted oTZ. The protocol actually admits two possible issuance protocols, or modes (cf. Figures 6 and 7): in free-mode, part of the statement (the argument $\boldsymbol{Y}$) is controlled by the user; in restricted-mode, the argument is selected by the server, possibly using some public information provided by the user. This results formally in two protocols that, when composed with a group generator GrGen, we denote $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ and $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$. The modality used depends on the concrete real-world scenario and allows to have more exact security guarantees.

All relations R have associated a setup algorithm Setup that takes as input $\Gamma = (\mathbb{G}, p, G) \leftarrow \mathsf{GrGen}(1^\lambda)$ as input and outputs a relation parameter Λ, which determines M. For restricted-mode, an additional (randomized) algorithm $\mathsf{SampleArg}$ is defined for R, which takes $(\mathit{crs}, \mathit{info})$ and randomness γ as input and outputs a row vector $\boldsymbol{Y} \in \mathbb{G}^{1\times m}$, and $\mathsf{Args}(\mathsf{R}, \mathit{info}) := \{v\boldsymbol{Y} : v \in \mathbb{Z}_p, \gamma \leftarrow\!\!\$\ \mathcal{R}, Y \leftarrow \mathsf{SampleArg}(\mathit{crs}, \mathit{info}; \gamma)\}$, where $\mathcal{R}$ denotes the space of the randomness.

The protocol is identical to non-interactive Σ-protocols described in the previous section, except that the prover additionally commits to another challenge that is used to re-randomized the one produced by the random oracle. We illustrate setup and verification procedures in Figure 5. Given $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z) \in \mathsf{AlgR}_{\Gamma,M}$ proving algorithm $\mathsf{oTZ.Prv}(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, Z)$ samples $a, b \leftarrow\!\!\$\ \mathbb{Z}_p$, $\boldsymbol{t} \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and computes $C := aH + bG$, $\boldsymbol{T} := M\boldsymbol{t}$. Then, computes the challenge $e := \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$ and returns $(e, a, b, \boldsymbol{r} := \boldsymbol{t} + ea\boldsymbol{x})$. Our oblivious issuance protocol with two modes for an algebraic relation R is showed in Figure 5 (the setup and verification algorithms) and Figures 6 and 7 (the issuing protocol, in two possible configurations: free and restricted).

Completeness of the protocol is immediate; knowledge-soundness follows special soundness of the underlying Σ-protocol. For instance, note that the protocol described in the random oracle model is also computationally special sound: given two transcripts $(\boldsymbol{T}, C, e, a, b, \boldsymbol{r})$ and $(\boldsymbol{T}, C, e', a', b', \boldsymbol{r}')$, then $a = a'$ is nonzero (by the binding property of the commitment scheme and verification equation), and using the canonical Σ-protocol extractor for the morphism $[\boldsymbol{Y}; M]$ using transcripts $(\boldsymbol{T}, ae, \boldsymbol{r})$ and $(\boldsymbol{T}', a'e', \boldsymbol{r}')$ it is possible the extract the witness $\boldsymbol{x}$. A transcript $(\boldsymbol{T}, C, e, a, b, \boldsymbol{r})$ can be simulated sampling $a, b, e, \boldsymbol{r}$ uniformly at random, and computing $\boldsymbol{T} = [\boldsymbol{Y}; M]\boldsymbol{r} - ea\boldsymbol{X}$ and $C := aH + bG$. Below, correctness of the issuance protocol.

Lemma 1. *The protocol* oTZ *is correct (in either mode).*

Proof. With overwhelming probability, $\upsilon \neq 0$. Define

$$R := \begin{bmatrix} \upsilon & 0 \\ 0 & I_{n+1} \end{bmatrix}\ .$$

Note that $[\boldsymbol{Y}'; M] = R[\boldsymbol{Y}; M]$. By definition, we have:

$$\begin{aligned} Y' &= \upsilon Y & \qquad e' &= \varepsilon\alpha^{-1}e \\ \boldsymbol{r}' &= \varepsilon\boldsymbol{r} + \boldsymbol{\rho} & a' &= \alpha a \\ \boldsymbol{T}' &= \varepsilon R\boldsymbol{T} + [\boldsymbol{Y}'; M]\boldsymbol{\rho} \end{aligned}$$

```
$\mathsf{oTZ.Setup}(1^\lambda)$
    $(\mathbb{G}, p, G) \coloneqq \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
    $H \leftarrow\!\!\$\ \mathbb{G}$
    $\Lambda \leftarrow \mathsf{R.Setup}(\Gamma)$
    return $crs \coloneqq (\Gamma, H, \Lambda)$

$\mathsf{oTZ.Ver}(crs, \boldsymbol{X}, \boldsymbol{Y}, Z, \pi = (a, b, e, \boldsymbol{r}))$
    $\boldsymbol{T} \coloneqq [\boldsymbol{Y}; M] \cdot \boldsymbol{r} - ea[Z; \boldsymbol{X}]$
    $C \coloneqq aH + bG$
    check $a \neq 0$ and $\boldsymbol{Y} \neq 0$
    return $e = \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$
```

Fig. 5. Setup and verification algorithms for our oblivious zero-knowledge protocol $\mathsf{oTZ}[\mathsf{GrGen}]$ for relation R. H is a random oracle with range in $\mathbb{Z}_p$.

Therefore, from the definition of $\boldsymbol{r}'$ as computed by Iss

$$\begin{aligned}
&\boldsymbol{r}' = \boldsymbol{t}' + e'a' \cdot \boldsymbol{x} \\
\Longrightarrow\ &[\boldsymbol{Y}'; M]\boldsymbol{r}' = [\boldsymbol{Y}'; M]\,(\boldsymbol{t}' + e'a'\boldsymbol{x}) \\
\Longrightarrow\ &[\boldsymbol{Y}'; M]\boldsymbol{r}' = \boldsymbol{T}' + e'a' \cdot [\boldsymbol{Y}'; M] \cdot \boldsymbol{x} \\
\Longrightarrow\ &R[\boldsymbol{Y}; M](\varepsilon\boldsymbol{r} + \boldsymbol{\rho}) = \varepsilon R\boldsymbol{T} + R[\boldsymbol{Y}; M]\boldsymbol{\rho} + (\varepsilon\alpha^{-1}e)(\alpha a)R[\boldsymbol{Y}; M] \cdot \boldsymbol{x} \\
\Longrightarrow\ &[\boldsymbol{Y}; M]\boldsymbol{r} = \boldsymbol{T} + ea \cdot [\boldsymbol{Y}; M] \cdot \boldsymbol{x}
\end{aligned}$$

which is the verification equation for $(\boldsymbol{Y}, Z), (a, b, e, \boldsymbol{r})$.

4.3 Security

For security, we only consider *simple* algebraic relations, which are defined as follows.

Definition 1. *An algebraic relation* R *is* simple *if there exists an efficient algorithm* Setup' *that takes* $\Gamma = (\mathbb{G}, p, G)$ *as input and outputs* Λ *together with a trapdoor td such that the DL of each entry of* M *to base* G *can be efficiently computed given td and the distribution of* Λ *is identical to that of the original setup algorithm* Setup.

In restricted mode, we additionally require there exists an efficient algorithm $\mathsf{SampleArg}'$ *that takes* $(crs, info, td)$ *as inputs and outputs* $\boldsymbol{Y} \in \mathbb{G}^{1\times m}$ *together with its DL* $\boldsymbol{y} \in \mathbb{Z}_p^m$ *to base* G *such that the distribution of* $\boldsymbol{Y}$ *is identical to that of* Setup *given* $(\Lambda, info)$ *as input.*

In addition to the standard requirements of soundness, we show the protocol is one-more unforgeable and oblivious.

One-More Unforgeability. We show that (i) protocol $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ is one-more unforgeable for the DLEQ relation $\mathsf{R}_{\mathrm{dleq}}$, and (ii) protocol $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ is one-more unforgeable for any *simple* algebraic relations where the kernel matrix Diffie-Hellman problem (KMDH) (defined in Section 2) is hard for M with distribution D_{R}, where $\mathsf{D}_{\mathsf{R}}(\Gamma)$ denotes the distribution of M after sampling $\Lambda \leftarrow \mathsf{R.Setup}(\Gamma)$. For the relations we study in Sections 6 and 7, this reduces to the DL assumption. Below, q_h denotes the maximum number of random oracle queries and q denotes the maximum number of signing queries to Iss_1 in the OMUF game.

Theorem 3. *If the* $(q+1)$*-SDL assumption is hard for* GrGen*, the protocol* $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ *for the DLEQ relation* $\mathsf{R}_{\mathrm{dleq}}$ *is one-more unforgeable in the algebraic group model and the random oracle model with advantage*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ}[\mathsf{GrGen},\mathrm{free}],\mathsf{R}_{\mathrm{dleq}}}(\lambda) \leq \mathsf{Adv}^{(\mathrm{q}+1)\text{-}\mathrm{sdl}}_{\mathsf{GrGen}}(\lambda) + \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \frac{(q_h + 3q)^2}{p-1} \ .$$

Theorem 4. *If DL is hard for* GrGen *and KMDH is hard for* GrGen *and* R*, the protocol* $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ *is one-more unforgeable in the algebraic group model and the random oracle model with advantage*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ}[\mathsf{GrGen},\mathrm{restr}],\mathsf{R}}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda) + \mathsf{Adv}^{\mathrm{kmdh}}_{\mathsf{GrGen},\mathsf{D}_{\mathsf{R}}}(\lambda) + \frac{(q_h + 3q)^2}{p-1} \ .$$

```
$\mathsf{oTZ.Usr}(crs, \boldsymbol{X}, info)$                                          $\mathsf{oTZ.Iss}(crs, \boldsymbol{x}, \boldsymbol{X})$

                                         $\xrightarrow{info}$
                                                                  $\boldsymbol{Y}' \leftarrow \mathsf{R.SampleArg}(crs, info)$
                                                                  $Z' := \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$
                                                                  $a' \leftarrow\!\!\$\ \mathbb{Z}_p^*, b' \leftarrow\!\!\$\ \mathbb{Z}_p$
                                                                  $C' := a'H + b'G$
                                                                  $\boldsymbol{t}' \leftarrow\!\!\$\ \mathbb{Z}_p^m$
                                                                  $\boldsymbol{T}' := \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \cdot \boldsymbol{t}'$
                                  $\xleftarrow{\boldsymbol{Y}', Z', \boldsymbol{T}', C'}$
$\upsilon \leftarrow\!\!\$\ \mathbb{Z}_p;\ \boldsymbol{Y} := \upsilon^{-1}\boldsymbol{Y}'$
$Z := \upsilon^{-1}Z'$
$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*, \beta \leftarrow\!\!\$\ \mathbb{Z}_p$
$C := \alpha^{-1}C' - \beta G$
$\varepsilon \leftarrow\!\!\$\ \mathbb{Z}_p^*;\ \boldsymbol{\rho} \leftarrow\!\!\$\ \mathbb{Z}_p^m$
$\boldsymbol{T} := \varepsilon^{-1} \begin{bmatrix} \upsilon^{-1} & 0 \\ 0 & I_n \end{bmatrix} \left( \boldsymbol{T}' - \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \boldsymbol{\rho} \right)$
$e := \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$
$e' := \varepsilon\, \alpha^{-1} e$
                                         $\xrightarrow{e'}$
                                                                  $\boldsymbol{r}' := \boldsymbol{t}' + e' a' \cdot \boldsymbol{x}$
                                    $\xleftarrow{r', a', b'}$
**check** $C' = a'H + b'G$ **and** $a' \neq 0$
$a := \alpha^{-1}a'$
$b := \alpha^{-1}b' - \beta$
**check** $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}' = e' a' [Z'; \boldsymbol{X}] + \boldsymbol{T}'$
$\boldsymbol{r} := \varepsilon^{-1}(\boldsymbol{r}' - \boldsymbol{\rho})$
**return** $((\boldsymbol{Y}, Z), \pi := (a, b, e, \boldsymbol{r}))$
```

Fig. 6. Oblivious issuance for the protocol $\mathsf{oTZ}[\mathsf{GrGen}, \mathsf{restr}]$ for relation R in **restricted** mode (that is, issuer chooses the full statement). In orange, the ROS mitigation inspired from Tessaro and Zhu [43] for concurrent security. In blue, the core differences with free issuance. Remaining, a natural extension of blind Schnorr for representation protocols, with multiplicative blinding for the challenge.

We first show the following lemma before proving the above two theorems.

Lemma 2. *In either mode, for any p.p.t. adversary* A *for the game* $\mathrm{OMUF}_{\mathsf{oTZ,R,A}}(\lambda)$ *making* q *queries to* $\textsc{Issuer}_1$*, there exists a p.p.t. adversary* B *for the game* $\mathrm{OMUF}_{\mathsf{oTZ,R,A},q}(\lambda)$ *such that all* q *signing sessions are finished and*

$$\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ,R,A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ,R,B}}(\lambda).$$

Proof. We construct a p.p.t. adversary B that runs A by forwarding all oracle queries from A to its own oracles. Assume (without loss of generality) that A makes making q queries to Iss_1 and $q + q_h$ queries to the random oracle H. After A returns, suppose $\ell < q$. We know $|Opn| = q - \ell$. Assume, without loss of generality, that the open session identifiers are $Opn = \{\ell + 1, \dots, q\}$. B internally runs the adversary A, forwarding all signing queries to the challenger. At the end, the adversary returns (Y_j^*, Z_j^*) and π_j^* for $j \in [\ell + 1]$. Also, denote them as $\{(Y_j, Z_j), \pi_j\}_{j \in [\ell+1]}$. For each unfinished session $i \in Opn$, B interacts with the challenger as described in Usr_1 and Usr_2 in restricted mode (cf. Figure 8), obtaining a new valid proof $\pi_j = (a_j, b_j, e_j, \boldsymbol{r}_j)$ for (Y_j, Z_j) and $j \in [\ell + 2, q + 1]$. If $Y_j = Y_{j'}$ for some $j' < j$, repeat this step

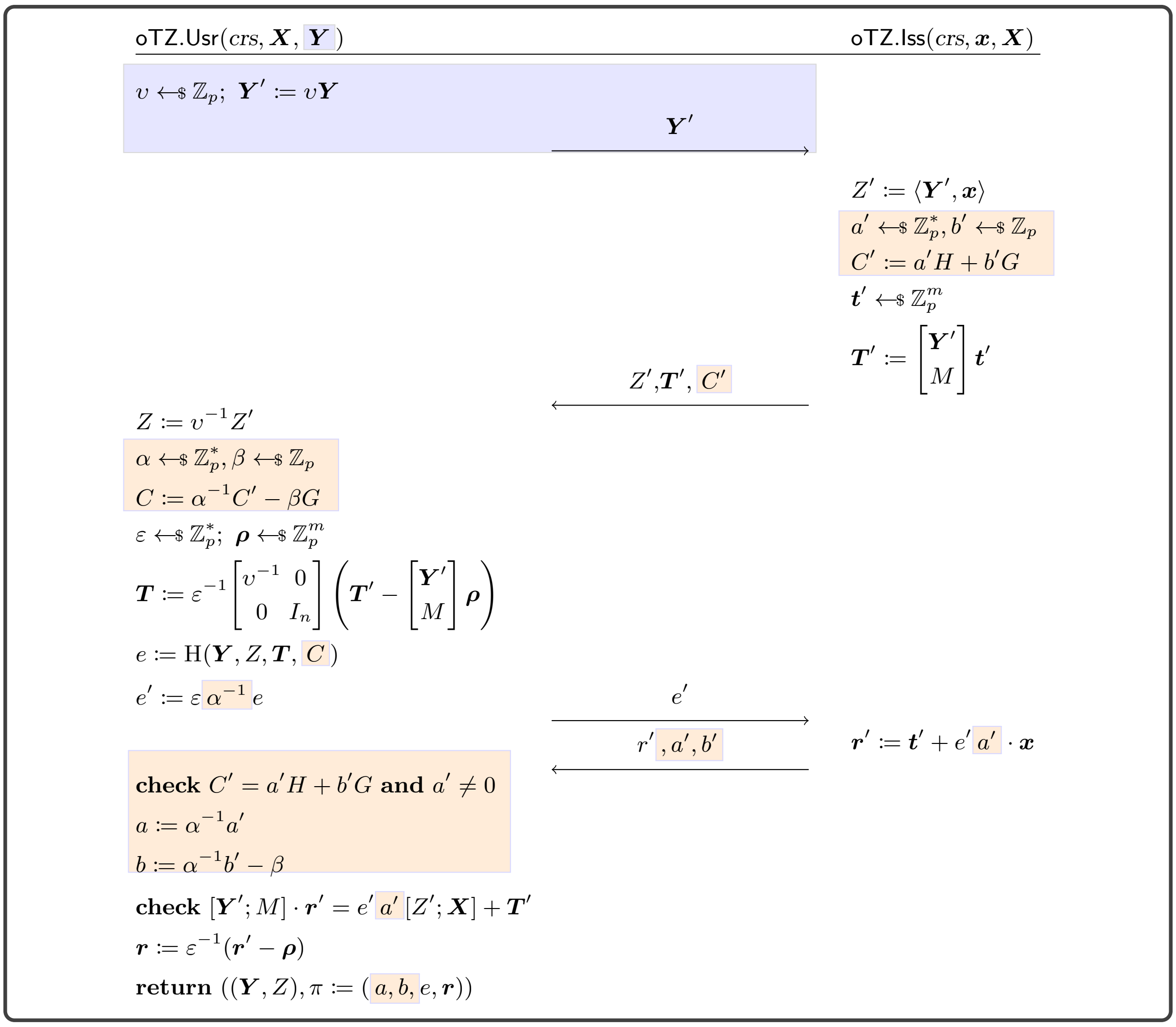
$\mathsf{oTZ.Usr}(crs, \boldsymbol{X}, \boldsymbol{Y})$
$\mathsf{oTZ.Iss}(crs, \boldsymbol{x}, \boldsymbol{X})$
$\upsilon \leftarrow\!\!\$\; \mathbb{Z}_p;\ \boldsymbol{Y}' := \upsilon \boldsymbol{Y}$
$\xrightarrow{\boldsymbol{Y}'}$
$Z' := \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$
$a' \leftarrow\!\!\$\; \mathbb{Z}_p^*, b' \leftarrow\!\!\$\; \mathbb{Z}_p$
$C' := a'H + b'G$
$\boldsymbol{t}' \leftarrow\!\!\$\; \mathbb{Z}_p^m$
$\boldsymbol{T}' := \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \boldsymbol{t}'$
$\xleftarrow{Z', \boldsymbol{T}', C'}$
$Z := \upsilon^{-1} Z'$
$\alpha \leftarrow\!\!\$\; \mathbb{Z}_p^*, \beta \leftarrow\!\!\$\; \mathbb{Z}_p$
$C := \alpha^{-1} C' - \beta G$
$\varepsilon \leftarrow\!\!\$\; \mathbb{Z}_p^*;\ \boldsymbol{\rho} \leftarrow\!\!\$\; \mathbb{Z}_p^m$
$\boldsymbol{T} := \varepsilon^{-1} \begin{bmatrix} \upsilon^{-1} & 0 \\ 0 & I_n \end{bmatrix} \left(\boldsymbol{T}' - \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \boldsymbol{\rho} \right)$
$e := \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$
$e' := \varepsilon \alpha^{-1} e$
$\xrightarrow{e'}$
$\boldsymbol{r}' := \boldsymbol{t}' + e' a' \cdot \boldsymbol{x}$
$\xleftarrow{\boldsymbol{r}', a', b'}$
check $C' = a'H + b'G$ **and** $a' \neq 0$
$a := \alpha^{-1} a'$
$b := \alpha^{-1} b' - \beta$
check $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}' = e' a' [Z'; \boldsymbol{X}] + \boldsymbol{T}'$
$\boldsymbol{r} := \varepsilon^{-1}(\boldsymbol{r}' - \boldsymbol{\rho})$
return $((\boldsymbol{Y}, Z), \pi := (a, b, e, \boldsymbol{r}))$

Fig. 7. Oblivious issuance for the protocol $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ for relation R in **free** mode, where the argument is under control of the user. In orange, the ROS mitigation inspired from Tessaro and Zhu [43] for concurrent security. In blue, the core differences with restricted issuance. Remaining, a natural extension of blind Schnorr for representation protocols, with multiplicative blinding for the challenge.

until $Y_j \neq Y_{j'}$ for any $j' < j$.[7] Using the valid responses obtained so far, B output $q+1$ a valid proofs $((Y_j, Z_j), \pi_j = (a_j, b_j, e_j, \boldsymbol{r}_j))_{j \in [q+1]}$.

Proof. (of Theorem 3 and Theorem 4) Since the proofs of the two theorems are very similar, we prove them together and highlight where the proofs differ. In free (respectively, restricted) mode, denote with A an algebraicadversary for $\mathrm{OMUF}_{\mathsf{oTZ}[\mathsf{GrGen},\mathrm{free/restr}],\mathsf{R},\mathsf{A}}(\lambda)$. By Lemma 2, we assume that A makes exactly q queries to $\textsc{Issuer}_1$ and $\ell = q$ (i.e., all sessions are finished at the end of A' s execution). Also, assume without loss of generality, that each $\left(\boldsymbol{Y}_j^*, Z_j^*, \pi_j^* = (a_j^*, b_j^*, e_j^*, \boldsymbol{r}_j^*)\right)$ output by A, a RO query $\mathrm{H}(\boldsymbol{Y}_j^*, Z_j^*, \boldsymbol{T}_j^*,$

[7] We remark that it's not really necessary for the challenger to blind the responses. However, this way B can find a υ_j such that $\upsilon_j Y_j' \neq Y_{j'}$ for any $j' < j$ at most repeating the protocol $(p-1)/(p-1-q)$ times, which is a constant.

```
oTZ.Setup($1^\lambda$)
    $(\mathbb{G}, p, G) := \Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$
    $H \leftarrow\!\$\ \mathbb{G}$
    $\Lambda \leftarrow \mathsf{R.Setup}(\Gamma)$
    return $crs := (\Gamma, H, \Lambda)$

$\dbox{oTZ.Usr_0(crs, \boldsymbol{X}, \boldsymbol{Y})}$
    $\upsilon \leftarrow\!\$\ \mathbb{Z}_p$
    $\boldsymbol{Y}' := \upsilon \boldsymbol{Y}$
    $st := (\boldsymbol{X}, \boldsymbol{Y}, \upsilon)$
    return $(st, \boldsymbol{Y}')$

oTZ.Iss$_1$($crs, \boldsymbol{x}, \dbox{\boldsymbol{Y}'}\ \boxed{info}$)
    $\boxed{\boldsymbol{Y}' \leftarrow \mathsf{R.SampleArg}(crs, info)}$
    $Z' := \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$
    $a' \leftarrow\!\$\ \mathbb{Z}_p^*, b' \leftarrow\!\$\ \mathbb{Z}_p$
    $C' := a'H + b'G$
    $\boldsymbol{t}' \leftarrow\!\$\ \mathbb{Z}_p^m$
    $\boldsymbol{T}' := \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \cdot \boldsymbol{t}'$
    $st := (\boldsymbol{x}, \boldsymbol{t}', a', b')$
    return $(st, (\boxed{\boldsymbol{Y}',} Z', \boldsymbol{T}', C'))$

oTZ.Usr$_1$($st, (\boxed{\boldsymbol{Y}',} Z', \boldsymbol{T}', C')$)
    $\dbox{(\boldsymbol{X}, \boldsymbol{Y}, \upsilon) := st}$
    $\boxed{\boldsymbol{X} := st}$
    $\boxed{\upsilon \leftarrow\!\$\ \mathbb{Z}_p;\ \boldsymbol{Y} := \upsilon^{-1}\boldsymbol{Y}'}$
    $Z := \upsilon^{-1} Z'$
    $\alpha \leftarrow\!\$\ \mathbb{Z}_p^*, \beta \leftarrow\!\$\ \mathbb{Z}_p$
    $C := \alpha^{-1} C' - \beta G$
    $\varepsilon \leftarrow\!\$\ \mathbb{Z}_p^*;\ \boldsymbol{\rho} \leftarrow\!\$\ \mathbb{Z}_p^m$
    $\boldsymbol{T} := \varepsilon^{-1} \begin{bmatrix} \upsilon^{-1} & 0 \\ 0 & I_n \end{bmatrix} \cdot \left( \boldsymbol{T}' - \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \boldsymbol{\rho} \right)$
    $e := \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$
    $e' := \varepsilon \alpha^{-1} e$
    $st := (\boldsymbol{X}, \boldsymbol{Y}, Z, \varepsilon, \boldsymbol{\rho}, \alpha, \beta)$
    return $(st, e')$

oTZ.Iss$_2$($st, e'$)
    $(\boldsymbol{x}, \boldsymbol{t}', a', b') := st$
    $\boldsymbol{r}' := \boldsymbol{t}' + e'a' \cdot \boldsymbol{x}$
    return $(\boldsymbol{r}', a', b')$

oTZ.Usr$_2$($st, (r', a', b')$)
    $(\boldsymbol{X}, \boldsymbol{Y}, Z, \varepsilon, \boldsymbol{\rho}, \alpha, \beta) := st$
    if $C' \neq a'H + b'G$ and $a' \neq 0$ :
        return $(\perp, \perp)$
    $a := \alpha^{-1} a'$
    $b := \alpha^{-1} b' - \beta$
    if $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}' \neq e'a'[Z'; \boldsymbol{X}] + \boldsymbol{T}'$ :
        return $(\perp, \perp)$
    $\boldsymbol{r} := \varepsilon^{-1}(\boldsymbol{r}' - \boldsymbol{\rho})$
    return $((\boldsymbol{Y}, Z), \pi := (a, b, e, \boldsymbol{r}))$

oTZ.Ver($crs, \boldsymbol{X}, \boldsymbol{Y}, Z, (a, b, e, \boldsymbol{r})$)
    $\boldsymbol{T} := [\boldsymbol{Y}; M] \cdot \boldsymbol{r} - ea[Z; \boldsymbol{X}]$
    $C := aH + bG$
    return $(e = \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C))$

$\boxed{oTZ.Usr_0(crs, \boldsymbol{X}, info)}$
    $st := (\boldsymbol{X})$
    return $(st, info)$
```

Fig. 8. The oblivious zero-knowledge proof protocol oTZ[GrGen] with two modes for the general discrete logarithm equality relation R, where GrGen denotes a group parameter generator. Free-mode (Y is chosen by the user) contains everything but the solid boxes. Restricted-mode (the user can only choose a public information *info*) contains everything but the dashed boxes. Notation is summarized in Table 1.

C_j^*) is made, where

$$\boldsymbol{T}_j^* := \begin{bmatrix} \boldsymbol{Y}_j^* \\ M \end{bmatrix} \boldsymbol{r}_j - e_j^* a_j^* \begin{bmatrix} Z_j^* \\ \boldsymbol{X} \end{bmatrix} , \tag{1}$$

$$C_j^* := b_j^* G + a_j^* H . \tag{2}$$

In restricted mode, without loss of generality, we assume $M_{1,1} \neq 0G$ and denote μ as the DL matrix of M which satisfies $M = \mu G$. Denote the k-th RO query as $(\boldsymbol{Y}_k, Z_k, \boldsymbol{T}_k, C_k)$. Since A is algebraic, A also provides the algebraic representations of the query with respect to all the group elements received by A. Denote the transcript of signing session i as $(\boldsymbol{Y}_i', Z_i', \boldsymbol{T}_i', C_i', e_i', r_i', a_i', b_i')$, and we have $Z_i' = \langle \boldsymbol{Y}_i', \boldsymbol{x} \rangle$ and $\boldsymbol{T}_i' = [\boldsymbol{Y}_i'; M]\boldsymbol{r}_i' - e_i' a_i' [Z_i'; \boldsymbol{X}]$.

Let $h := \log_G H$, $\mu := \log_G M$, and $\boldsymbol{y}_i' := \log_G \boldsymbol{Y}_i'$. We first show the following lemma .

Lemma 3. *For each RO query, we can represent*

$$\begin{aligned} T_{k,2} &= (\alpha_k(x_1) + \alpha_k'(x_1) \cdot h)G , \\ C_k &= (\beta_k(x_1) + \beta_k'(x_1) \cdot h)G , \end{aligned} \tag{3}$$

where $\alpha_j(\mathsf{X}), \alpha_j'(\mathsf{X}), \beta_j(\mathsf{X}), \beta_j'(\mathsf{X}) \in \mathbb{Z}_p[\mathsf{X}]$ *can be computed efficiently when* A *returns given* μ, $\{\boldsymbol{y}_i'\}_{i \in [q]}$ *(only in restricted mode), and* $(x_2, \dots, x_m)$*.*

Proof. In free mode for $\mathsf{R}_{\mathrm{dleq}}$, we have the length of $\boldsymbol{x}$ is 1 and therefore we just use x to denote x_1. We first show how to compute $\eta_i(\cdot), \eta'_i(\cdot)$ for each $i \in [q]$ such that

$$Y'_i = (\eta_i(x) + \eta'_i(x) \cdot h)G \; . \tag{4}$$

– For $i = 1$, since A is algebraic, we know $Y'_1 = \xi^{(G)}G + \xi^{(X)}X + \xi^{(H)}H$ where $\xi^{(G)}, \xi^{(X)}, \xi^{(H)}$ are constants given by A, and we let $\eta_1(\mathsf{X}) \coloneqq \xi^{(G)} + \xi^{(X)}\mathsf{X}$ and $\eta'_1(\mathsf{X}) \coloneqq \xi^{(H)}$.
– For $1 < i \leq q$, since A is algebraic, we know $Y'_i = \xi^{(G)}G + \xi^{(X)}X + \xi^{(H)}H + \sum_{j\in[i-1]}(\xi^{(T'_{j,1})}T'_{j,1} + \xi^{(T'_{j,2})}T'_{j,2} + \xi^{(C'_j)}C'_j + \xi^{(Y'_j)}Y'_j + \xi^{(Z'_j)}Z'_j)$, where $\xi^{(\cdot)}$ are given by A. Since $T'_{j,1} = (r'_j - e'_j a'_j x)Y'_j$, $T'_{j,2} = (r'_j - e'_j a'_j x)G$, $C'_j = (b'_j + a'_j h)G$, and $Z'_j = xY'_j$, we let

$$\begin{aligned}\eta_i(\mathsf{X}) \coloneqq \xi^{(G)} + \xi^{(X)}\mathsf{X} + \sum_{j\in[i-1]} \Big(&\xi^{(T_{j,2})'}(r'_j - e'_j a'_j \mathsf{X}) + \xi^{(C'_j)}b'_j \\ &+ (\xi^{(T'_{j,1})}(r'_j - e'_j a'_j \mathsf{X}) + \xi^{(Y'_j)} + \xi^{(Z'_j)}\mathsf{X})\eta_j(\mathsf{X})\Big) \; , \\ \eta'_i(\mathsf{X}) \coloneqq \xi^{(H)} + \sum_{j\in[i-1]} \Big(&\xi^{(C'_j)}a'_j + (\xi^{(T'_{j,1})}(r'_j - e'_j a'_j \mathsf{X})\xi^{(Y'_j)} + \xi^{(Z'_j)}\mathsf{X})\eta'_j(\mathsf{X})\Big) \; .\end{aligned} \tag{5}$$

Then, for the k-th RO query $(Y_k, Z_k, \boldsymbol{T}_k, C_k)$, since A is algebraic, we know a representation of $T_{k,2}$ and C_k as a linear combination of $G, X, H, \{T'_{i,1}, T'_{i,2}, C'_i, Y'_i, Z'_i\}_{i\in[q]}$. Thus, we can compute $\alpha_j(\cdot), \alpha'_j(\cdot), \beta_j(\cdot), \beta'_j(\cdot)$ using the same argument as in the previous step.

In restricted mode, since A is algebraic, for the k-th RO query, we know $T_{k,2}$ and C_k as a linear combination of $G, \boldsymbol{X}, H, \{\boldsymbol{T}'_i, C'_i, \boldsymbol{Y}'_i, Z'_i\}_{i\in[q]}$. Since $\boldsymbol{T}'_i = [\boldsymbol{Y}'_i; M](\boldsymbol{r}'_i - a'_i e'_i \boldsymbol{x})$, $C'_i = a'_i H + b'_i G$, $Z'_i = \langle \boldsymbol{Y}'_i, \boldsymbol{x}\rangle$, and we are given μ, $\{\boldsymbol{y}'_i\}_{i\in[q]}$ and $(x_2, \ldots, x_m)$, we can compute $\{\tilde{\alpha}_j, \tilde{\beta}_j\}_{j\in[q+3]} \in \mathbb{Z}_p$ such that

$$\begin{aligned} T_{k,2} &= (\tilde{\alpha}_1 + \tilde{\alpha}_2 x + \tilde{\alpha}_3 h)G + \sum_{i\in[q]} \tilde{\alpha}_{i+3}(r'_{i,1} - a'_i e'_i \boldsymbol{x})G \; , \\ C_k &= (\tilde{\beta}_1 + \tilde{\beta}_2 x + \tilde{\beta}_3 h)G + \sum_{i\in[q]} \tilde{\beta}_{i+3}(r'_{i,1} - a'_i e'_i \boldsymbol{x})G \; . \end{aligned} \tag{6}$$

Thus, we can compute $\alpha_j(\cdot), \alpha'_j(\cdot), \beta_j(\cdot), \beta'_j(\cdot)$ from $\{\tilde{\alpha}_j, \tilde{\beta}_j\}_{j\in[q+3]}$. □

We proceed by means of a hybrid argument.

Hyb_1 this is the original game, described above

Hyb_2 We replace the procedures $\mathsf{R.Setup}$ and $\mathsf{R.SampleArg}$ with (respectively) $\mathsf{R.Setup}'$ and $\mathsf{R.SampleArg}'$, which will provide μ and $\boldsymbol{y}'_i$ whenever it also outputs a new argument $\boldsymbol{Y}'_i$ during the i-th query. Since the relation is simple, this change is perfectly indistinguishable from the previous one.

Hyb_3 we strengthen the game and add another condition before returning. Let us index in $k_j \in [q_h]$ the random oracle query that the adversary makes associated with the j-th forgery. If, among the proofs returned by the adversary, $\exists j \in [q+1]$ such that $\beta'_{k_j}(x_1) \neq a^*_j$, the game immediately aborts and the adversary loses.

This hybrid is computationally indistinguishable from the first one and follows from the binding property of the commitment scheme. We consider an adversary B for the game $\mathrm{BIND}_{\mathsf{Com[GrGen]},\mathsf{B}}(\lambda)$ that, upon receiving a group description $\Gamma = (\mathbb{G}, p, G)$ and a commitment key H, computes $\Lambda \leftarrow R.\mathsf{Setup}(\Gamma)$, samples $\boldsymbol{x} \leftarrow\!\!\$\; \mathbb{Z}_p^m$, runs A on input $((\Gamma, H, \Lambda), X \coloneqq M\boldsymbol{x})$. B responds to the signing queries just as the challenger in the game $\mathrm{OMUF}_{\mathsf{oTZ,R,A}}(\lambda)$. Once the adversary returns, if $\exists j \in [q+1]$ such that $\beta'_{k_j}(x_1) \neq a^*_j$, then B outputs the commitment C_k along with two valid openings $(a_j, b_j), (\beta'_{k_j}(x_1), \beta_{k_j}(x_1))$. In fact, (a_j, b_j) are valid if the proof π^*_j verifies, while the second opening is correctly given by any algebraic adversary. Therefore, the distinguishing advantage between Hyb_2 and Hyb_3 is bounded by $\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda)$.

Hyb_4 we add one additional condition before returning. If there exists $j \in [q+1]$ such that $\mathrm{coe}_1(\alpha_{k_j}(\mathsf{X}) + \alpha'_{k_j}(\mathsf{X}) \cdot h) \neq -\mu_{1,1} e^*_j a^*_j$, where $\mathrm{coe}_1(f(\mathsf{X}))$ denotes the coeffcient of the first degree term X in polynomial f, the game immediately aborts and the adversary loses.

– In free mode for the DLEQ relation R_{dleq}, this hybrid is computationally indistinguishable from the previous if the $(q+1)$-strong DL assumption is hard for GrGen.
  We show the above by constructing an adversary B for (q + 1)-$\mathrm{SDL}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. Initially, B initializes $i \coloneqq 0$ and Opn as in the previous hybrid. After B receives a group description Γ together with a challenge $\boldsymbol{W} = (xG, x^2G, \ldots, x^{q+1}G)$ from the game (q + 1)-$\mathrm{SDL}_{\mathsf{GrGen},\mathsf{B}}(\lambda)$, B generates a commitment key $H \coloneqq hG$ and a trapdoor $h \leftarrow\!\!\$\ \mathbb{Z}_p$. Then, B runs A on input $((\Gamma, H, \Lambda), W_1)$ and replies to the oracle queries as follows:
  - *Oracle* $\textsc{Issuer}_1(i, Y_i')$*:* B samples $r_i', \hat{b}_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\hat{a}_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and sets $T_{i,2}' \coloneqq (r_i' - \hat{a}_i x)G = r_i'G - \hat{a}_i W_1$ and $C_i' \coloneqq \hat{b}_i G$. For computing $T_{i,1}'$ and Z_i, since A is algebraic, B will have a representation of Y_i' as a linear combination of $G, W_1, H, \{T_{k,1}', T_{k,2}', C_k', Y_k', Z_k'\}_{k\in[i-1]}$. For $i = 1$, since $H = hG$, the adversary B knows a representation of Y_i' as a linear combination of G, W_1. For $i > 1$, B knows a representation of Y_i' as a linear combination of $G, W_1, \ldots, W_i$, since for each $k \in [i-1]$, $T_{k,0}' = r_k'G - a_k'W_1$, $C_k' = (\hat{a}_k h + \hat{b}_k)G$, $T_{k,1}' = (r_k' - \hat{a}_k x)Y_k'$, $Z_k' = xY_k'$, and B knows a representation of Y_k' as a linear combination of $G, W_1, \ldots, W_k$, Therefore, B can compute $Z_i' \coloneqq xY_i'$ and $T_{i,1}' \coloneqq (r_i' - \hat{a}_k x)Y_i$ using $G, W_1, \ldots, W_{i+1}$. Finally, B returns $(Z_i', \boldsymbol{T}_i', C_i')$.
  - *Oracle* $\textsc{Issuer}_2(i, e_i')$*:* B equivocates the commitment setting $a_i' \coloneqq \hat{a}_i / e_i'$ and $b_i' \coloneqq \hat{b}_i - a_i' \cdot h$. Note that
  $$C_i ;= \hat{b}_i G = a_i' H + b_i' G.$$
  B returns (r_i', a_i', b_i').
  - *Oracle* H*:* Same as in the $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R}_{\mathrm{dleq}},\mathsf{A}}(\lambda)$: if the query has been previously formulated, reply with the same value, otherwise sample a uniformly random element from $\mathbb{Z}_p$ and return it.

– In restricted mode, this hybrid is computationally indistinguishable from the previous one if DL is hard for GrGen.
  We show the above by constructing an adversary B for $\mathrm{DL}_{\mathsf{B},\mathsf{GrGen}}(\lambda)$. The adversary B, upon receiving as input a group description Γ and a DL challenge $W \in \mathbb{G}$, it generates a trapdoor $h \leftarrow\!\!\$\ \mathbb{Z}_p$ and commitment key $H = hG$. Also, B runs $(\Lambda, td) \leftarrow \mathsf{R.Setup}'(\Gamma)$ and computes M with its DL matrix $\mu \in \mathbb{Z}_p^{n\times m}$ such that $M = \mu G$. It samples $x_2, \ldots, x_m \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets $\boldsymbol{X} = \mu \cdot [W; x_2G; \ldots; x_mG]$. It internally runs $\mathsf{A}((\Gamma, H, \Lambda), \boldsymbol{X})$ and replies to the oracles queries as follows:
  - *Oracle* $\textsc{Issuer}_1(i, \mathit{info})$*:* B runs $(\boldsymbol{Y}_i', \boldsymbol{y}_i') \leftarrow \mathsf{R.SampleArg}'(\mathit{crs}, \mathit{info})$, where $\boldsymbol{Y}_i' = \boldsymbol{y}_i'G$. Then, it computes $Z_i' \coloneqq \langle \boldsymbol{y}_i', [W; x_2G; \ldots; x_mG]\rangle$. B creates a valid (simulated) transcript: sample $\boldsymbol{r}_i' \leftarrow\!\!\$\ \mathbb{Z}_p$ and $\hat{a}_i \leftarrow\!\!\$\ \mathbb{Z}_p^*, \hat{b}_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes
  $$C_i' \coloneqq \hat{b}_i G$$
  $$\boldsymbol{T}_i' \coloneqq [\boldsymbol{Y}_i'; M]\boldsymbol{r}_i' - \hat{a}_i [Z_i'; \boldsymbol{X}]$$
  Finally, B returns $(\boldsymbol{Y}_i', Z_i', C_i', \boldsymbol{T}_i')$.
  - *Oracle* $\textsc{Issuer}_2(i, e_i')$ *and* H: Same as in free mode described above.

At the end of the execution of B, if $\exists j \in [q+1]$ such that $\mathrm{coe}_1(\alpha_{k_j}(\mathsf{X}) + \alpha_{k_j}'(\mathsf{X})) \neq -\mu_{1,1} e_j^* a_j^* \mathsf{X}$. Since $(\alpha_{k_j}(x_1) + \alpha_{k_j}'(x_1) \cdot h)G = T_{j,2}^* = \sum_{i'=1}^m M_{1,i'}(r_{j,i'}^* - e_j^* a_j^* x_{i'})$, B can computes x_1, which is the discrete logarithm of the challenge, by computing one of the roots of the non-zero polynomial $f(\mathsf{X}) = \alpha_{k_j}(\mathsf{X}) + \alpha_{k_j}'(\mathsf{X}) \cdot h - \sum_{i'=2}^m \mu_{1,i'}(r_{j,i'}^* - e_j^* a_j^* x_{i'}) - \mu_{1,1} r_{j,1}^* + \mu_{1,1} e_j^* a_j^* \mathsf{X}$. Note that the above analysis directly works for the DLEQ relation R_{dleq} in free-mode too, where we have $m = 1$, $x = x_1$, and $\mu_{1,1} = 1$. Therefore, in free mode, the distinguishing advantage between Hyb_3 and Hyb_4 is bounded by $\mathsf{Adv}_{\mathsf{GrGen}}^{(\mathrm{q}+1)\text{-sdl}}(\lambda)$, and in restricted mode, the advantage is bounded by $\mathsf{Adv}_{\mathsf{GrGen}}^{\mathrm{dl}}(\lambda)$.

Hyb_5 we add another condition before returning. We first define introduce some notation and a lemma before defining the condition. Denote Opn_k the set of sessions open (i.e., a query to $\textsc{Issuer}_1$ for a session i was made but no query to $\textsc{Issuer}_2$ for i was made yet) during the k-th random oracle query. Denote $\hat{b}_i$ as the DL of C_i' to base G and $\boldsymbol{t}_i'$ as the DL of $\boldsymbol{T}_i'$ to base G. Denote a polynomial $P_S(\mathsf{X}) \coloneqq \prod_{i\in S}(r_{i,1}' - e_i' a_i' \mathsf{X})$ for each $S \subseteq [q]$. In particular, $P_\emptyset(\mathsf{X}) \coloneqq 1$.

Lemma 4. *For each $j \in [q_h]$, $\alpha_j(\cdot), \alpha'_j(\cdot), \beta'_j(\cdot)$ have the following form*

$$\alpha_j(\mathsf{X}) + \alpha'_j(\mathsf{X}) \cdot h = \sum_{S \subseteq Opn_j} \tilde{\alpha}_j^{(S)}(\mathsf{X}) P_S(\mathsf{X}) ,$$

$$\beta'_j(x_1) = \tilde{\beta}_j^{(0)} + \sum_{i \in [q]} \tilde{\beta}_j^{(i)} a'_i ,$$

where each $\tilde{\beta}_j^{(i)}$ and the coefficients of each $\alpha_j^{(S)}(\mathsf{X})$ can be efficiently computed when the j-th RO query is made given μ, $\{\boldsymbol{y}'_i\}_{i\in[q]}$ (only in restricted mode), $(x_2, \ldots, x_m)$, h, and $\{\boldsymbol{t}'_i, \hat{b}_i\}_{i\in[q]}$.

Proof. In restricted mode, we know the lemma holds by Equation (6). In free mode for $\mathsf{R}_{\mathrm{dleq}}$, we first show how to compute $\hat{\eta}_i(\cdot), \tilde{\eta}_i$ for each $i \in [i-1]$ such that

$$\eta_i(\mathsf{X}) + \eta'_i(\mathsf{X}) \cdot h = \sum_{S \subseteq [i-1]} \hat{\eta}_i^{(S)}(\mathsf{X}) P_S(\mathsf{X}) ,$$

$$\eta'_i(x) = \tilde{\eta}_i^{(0)} + \sum_{j \in [i-1]} \tilde{\eta}_i^{(j)} a_j ,$$

where η_i and η'_i from Equation (4).

- For $i = 1$, since $\eta_1(\mathsf{X}) = \xi^{(G)} + \xi^{(X)} X$ and $\eta'_1(\mathsf{X}) = \xi^{(H)}$ where $\xi^{(G)}, \xi^{(X)}, \xi^{(H)}$ are constants given by A, we let $\hat{\eta}^{(\emptyset)}(\mathsf{X}) := \xi^{(G)} + \xi^{(H)} h + \xi^{(X)} \mathsf{X}$ and $\tilde{\eta}_1^{(0)} := \xi^{(H)}$. All other $\hat{\eta}^{(S)}$ and $\tilde{\eta}_1^{(i)}$ are set to 0.
- For $1 < i \leq q$, by Equation (5), since $t'_{i'} = r'_{i'} - e'_{i'} a'_{i'}$ and $\hat{b}_{i'} = b_{i'} + a_{i'} h$ for each $i' \in [q]$, we let

$$\hat{\eta}_i^{(S)}(\mathsf{X}) := \begin{cases} \xi^{(G)} + \xi^{(X)}\mathsf{X} + \xi^{(H)} h + \sum_{i'\in[i-1]} \Big(\xi^{(C'_{i'})} \hat{b}_{i'} + (\xi^{(Y'_{i'})} + \xi^{(Z'_{i'})}\mathsf{X}) \hat{\eta}_{i'}^{(\emptyset)}(\mathsf{X}) \Big) , \text{ for } S = \emptyset \\ \xi^{(T'_{j,2})} + \xi^{(T'_{j,1})} \hat{\eta}_j^{(\emptyset)}(\mathsf{X}) + \sum_{i'\in[i-1]} (\xi^{(Y'_{i'})} + \xi^{(Z'_{i'})}\mathsf{X}) \hat{\eta}_{i'}^{(S)}(\mathsf{X}) , \text{ for } S = \{j\}, j \in [i-1] \\ \sum_{i'\in[i-1]} (\xi^{(Y_{i'})} + \xi^{(Z_{i'})}\mathsf{X}) \hat{\eta}_j^{(S)}(\mathsf{X}) + \sum_{i' \in S} \xi^{(T'_{i',1})} \hat{\eta}_{i'}^{(S \setminus \{i'\})}(\mathsf{X}) , \text{ for } S \subseteq [i-1], |S| > 1 \\ 0 , \ o.w. \end{cases} ,$$

$$\tilde{\eta}_i^{(k)} := \begin{cases} \xi^{(H)} + \sum_{i'\in[i-1]} (\xi^{(T'_{i',1})} t_{i'} + \xi^{(Y'_{i'})} + \xi^{(Z'_{i'})} x) \tilde{\eta}_{i'}^{(0)} , & j = 0 \\ \xi^{(C_j)} + \sum_{i'\in[i-1]} (\xi^{(T'_{i',1})} t_{i'} + \xi^{(Y'_{i'})} + \xi^{(Z'_{i'})} x) \tilde{\eta}_{i'}^{(j)} , & j \in [i] \\ 0 , & o.w. \end{cases} .$$

Since we know a representation of $T_{k,2}$ and C_k as a linear combination of $G, X, H, \{T_{i,1}, T_{i,2}, C_i, Y_i, Z_i\}_{i\in[q]}$, we can compute $\bar{\alpha}_k$ and $\tilde{\beta}_k$ using a similar way as computing $\hat{\eta}$ and $\tilde{\eta}$ in the previous step, where $\bar{\alpha}_k$ satisfies $\alpha_k(\mathsf{X}) + \alpha'_k(\mathsf{X}) \cdot h = \sum_{S\subseteq[q]} \bar{\alpha}_k^{(S)}(\mathsf{X}) P_S(\mathsf{X})$. Since for each $i \in [q] \setminus Opn_k$, the values r'_i, e'_i, a'_i are known when the k-the RO query is made, we compute $\tilde{\alpha}_j^{(S)}(\mathsf{X}) := \sum_{S' \subseteq [q] \setminus Opn_j} \bar{\alpha}_j^{(S' \cup S)}(\mathsf{X}) \prod_{i \in S'} (r'_i - e'_i a'_i \mathsf{X})$ for each $S \subseteq Opn_k$. □

After A returns, the game aborts and A immediately loses, if there exists $j \in [q+1]$ such that $\alpha_{k_j}(\cdot), \alpha'_{k_j}(\cdot)$ does not satisfy $\alpha_{k_j}(\mathsf{X}) + \alpha'_{k_j}(\mathsf{X}) \cdot h = \tilde{\alpha}_{k_j}^{(\emptyset)}(\mathsf{X}) + \sum_{i \in Opn_{k_j}} \tilde{\alpha}_{k_j,0}^{(\{i\})} \cdot (r'_{i,0} - e'_i a'_i \mathsf{X})$, where $\tilde{\alpha}_{k_j,0}^{(\{i\})}$ denotes the constant term of the polynomial $\tilde{\alpha}_{k_j}^{(\{i\})}(\mathsf{X})$. This hybrid change is indistinguishable from the previous hybrid by constructing a p.p.t. adversary B that wins $\mathrm{WFROS}_{q,p,\mathsf{B}}(\lambda)$ (defined in Figure 1) every time that A wins in Hyb_4 but not in Hyb_5.
We now show how B is constructed. Initially, B initializes $i := 0, Opn := \emptyset$ as described in the $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R},\mathsf{A}}(\lambda)$ game. B runs $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and $(\Lambda, td) \leftarrow \mathsf{R.Setup}'(\Gamma)$. B samples $\boldsymbol{x} \leftarrow\$ \mathbb{Z}_p^m$. Then, it creates a new commitment key by sampling $h \leftarrow\$ \mathbb{Z}_p$ and setting $H := hG$. B runs A on input $((\Gamma, H, \Lambda), \boldsymbol{X} := M\boldsymbol{x})$. It replies to each oracle query using its own oracles S and $\mathrm{H}_{\mathrm{wfros}}$ as follows:

- *Oracle* $\textsc{Issuer}_1(i, \boldsymbol{Y}'_i)$ *in free mode:* same as the $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R}_{\mathrm{dleq}},\mathsf{A}}(\lambda)$ game except B samples a new variable $\hat{b}_i \leftarrow\$ \mathbb{Z}_p$, and sets $C'_i = \hat{b}_i G$.

- *Oracle* $\textsc{Issuer}_1(i, \mathit{info})$ *in restricted mode:* same as the $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R},\mathsf{A}}(\lambda)$ game except B runs $(\boldsymbol{Y}'_i, \boldsymbol{y}'_i) \leftarrow \mathsf{R.SampleArg}(\mathit{crs}, \mathit{info}, \mathit{td})$ to generate $\boldsymbol{Y}'_i$ and samples a new variable $\hat{b}_i \leftarrow\!\$\ \mathbb{Z}_p$, and sets $C'_i = \hat{b}_i G$.
- *Oracle queries* $\textsc{Issuer}_2(i, e'_i)$*:* if $i \notin \mathit{Opn}$ or $e'_i = 0$, B returns $\perp$. Otherwise, B makes a query (i, e'_i) to S and uses its output as the value a'_i. Also, B sets $b'_i = \hat{b}_i - a'_i \cdot h$. With the value $(\boldsymbol{r}'_i, a'_i, b'_i)$, the rest of $\textsc{Issuer}_2$ proceeds as Iss_2 in the $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R},\mathsf{A}}(\lambda)$ game.
- *Oracle queries* $\mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$*:* if $\mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C) \neq \perp$, the value $\mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$ is returned. Otherwise, suppose this is the k-th query. B computes $\tilde{\alpha}^{(\emptyset)}_{k,1}, \{\tilde{\alpha}^{(\{i\})}_{k,0}\}_{i \in \mathit{Opn}_k}$ and $\{\tilde{\beta}^{(i)}_k\}_{i \in \{0\} \cup [q]}$ where $\tilde{\alpha}^{(S)}_k$ and $\tilde{\beta}^{(i)}_k$ are defined in Lemma 4 and $\tilde{\alpha}^{(S)}_{k,i'}$ represents the coefficient of $\mathsf{X}^{i'}$ in $\tilde{\alpha}^{(S)}_k(\mathsf{X})$. Then, B issues the query $(\hat{\boldsymbol{\alpha}}_k, \hat{\boldsymbol{\beta}}_k, k)$ to $\mathrm{H}_{\mathrm{wfros}}$, where $\hat{\boldsymbol{\alpha}}_j, \hat{\boldsymbol{\beta}}_j \in \mathbb{Z}_p^{q+1}$ are such that

$$\begin{aligned} \hat{\alpha}_{k,i} &= \begin{cases} \tilde{\alpha}^{(\emptyset)}_1 / \mu_{1,1}\ , & i = 0 \\ -\tilde{\alpha}^{(\{i\})}_0 / \mu_{1,1}\ , & i \in \mathit{Opn}_j \\ 0\ , & o.w. \end{cases}\ , \\ \hat{\beta}_{k,i} &= -\tilde{\beta}^{(i)}_k\ ,\ \text{ for } i = 0, \ldots, q\ . \end{aligned} \tag{7}$$

After receiving the output δ_k, B returns $\mathrm{H}(\boldsymbol{Y}_k, Z_k, \boldsymbol{T}_k, C_k) \coloneqq \delta_k$.

After A returns, B outputs $\{(\hat{\boldsymbol{\alpha}}_{k_j}, \hat{\boldsymbol{\beta}}_{k_j}, k_j)\}_{j \in [q+1]}$, where k_j is defined in Hyb_3.
We now show B wins if A wins in Hyb_4 but not in Hyb_5. We first show for any $j_1 \neq j_2 \in [q+1]$, $(\hat{\boldsymbol{\alpha}}_{k_{j_1}}, \hat{\boldsymbol{\beta}}_{k_{j_1}}, k_{j_1}) \neq (\hat{\boldsymbol{\alpha}}_{k_{j_2}}, \hat{\boldsymbol{\beta}}_{k_{j_2}}, k_{j_2})$. Suppose there exists $j_1 \neq j_2 \in [q+1]$ such that $k_{j_1} = k_{j_2}$, which implies $(e^*_{j_1}, \boldsymbol{Y}^*_{j_1}, Z^*_{j_1}, \boldsymbol{T}^*_{j_1}, C^*_{k_1}) = (e^*_{j_2}, \boldsymbol{Y}^*_{j_2}, Z^*_{j_2}, \boldsymbol{T}^*_{j_2}, C^*_{j_2})$ are the same RO query. From the previous hybrids, we have $a^*_{j_1} = \beta'_{k_{j_1}}(x) = \beta'_{k_{j_2}}(x) = a^*_{j_2}$, which also implies $b^*_{j_1} = b^*_{j_2}$. Since $[\boldsymbol{Y}^*_{j_1}; M]\boldsymbol{r}^*_{j_1} = \boldsymbol{T}^*_{j_1} - e^*_{j_1} y^*_{j_1} \boldsymbol{X} = \boldsymbol{T}^*_{j_2} - e^*_{j_2} y^*_{j_2} \boldsymbol{X} = [\boldsymbol{Y}^*_{j_2}; M]\boldsymbol{r}^*_{j_2}$, we have $M\boldsymbol{r}^*_{j_1} = M\boldsymbol{r}^*_{j_2}$. In free mode for R_{dleq}, we have $r^*_{j_1} = r^*_{j_2}$. In retricted mode, $r^*_{j_1} = r^*_{j_2}$ is implied by the kernel Diffie-Hellman hardness of R. Thus, we have $(\boldsymbol{Y}^*_{j_1}, Z^*_{j_1}, \pi^*_{j_1}) = (\boldsymbol{Y}^*_{j_2}, Z^*_{j_2}, \pi^*_{j_2})$, which contradicts with the winning condition of A. Since $e^*_{k_1} = \delta_{j_{k_1}} = \delta_{j_{k_1}} = e^*_{k_2}$, we have $(Y^*_{k_1}, Z_{k^*_1} = xY^*_{k_1}, \pi^*_{k_1} = (r^*_{k_1}, e^*_{k_1}, a^*_{k_1}, b^*_{k_1})) = (Y^*_{k_2}, Z_{k^*_2} = xY^*_{k_2}, \pi^*_{k_2} = (r^*_{k_2}, e^*_{k_2}, a^*_{k_2}, b^*_{k_2}))$, which contradicts the winning condition of A.
Also, since A wins in Hyb_4 but not in Hyb_5, we have for each $j \in [q+1]$,

$$\begin{aligned} -\mu_{1,1} e^*_k a^*_k &= \mathrm{coe}_1(\alpha_{k_j}(\mathsf{X}) + \alpha'_{k_j}(\mathsf{X}) \cdot h) \\ &= \mathrm{coe}_1\left(\tilde{\alpha}^{(\emptyset)}_{k_j}(\mathsf{X}) + \sum_{i \in \mathit{Opn}_{k_j}} \tilde{\alpha}^{(\{i\})}_{k_j}(\mathsf{X})(r'_{i,1} - e'_i a'_i \mathsf{X})\right) \\ &= \tilde{\alpha}^{(\emptyset)}_{k_j,1} - \sum_{i \in \mathit{Opn}_{k_j}} \tilde{\alpha}^{(\{i\})}_{k_j,0} e'_i a'_i\ , \end{aligned}$$

$$\tilde{\beta}^{(0)}_{j_k} + \sum_{i \in [q]} \tilde{\beta}^{(i)}_{j_k} a'_i = a^*_k\ \ ,$$

which implies $\hat{\alpha}_{k_j,0} + \sum_{i \in [q]} \hat{\alpha}_{k_j,i} e'_i a'_i = e^*_j(\hat{\beta}_{k_j,0} + \sum_{i \in [q]} \hat{\beta}_{k_j,i} a'_i)$. Since $e^*_k = \mathrm{H}(\boldsymbol{Y}^*_k, Z^*_k, \boldsymbol{T}^*_k, C^*_k) = \mathrm{H}_{\mathrm{wfros}}(\hat{\boldsymbol{\alpha}}_{k_j}, \hat{\boldsymbol{\beta}}_{k_j}, k_j)$, B wins the game $\mathrm{WFROS}_{q,p,\mathsf{B}}(\lambda)$. By Theorem 2, the distinguishing advantage between Hyb_4 and Hyb_5 is bounded by $\frac{q_h(2q+q_h)}{p-1}$ in free mode, and is bounded by $\mathsf{Adv}^{\mathrm{kmdh}}_{\mathsf{GrGen},\mathsf{D}}(\lambda) + \frac{q_h(2q+q_h)}{p-1}$ in restricted mode.

Finally, we can conclude the theorems by Lemma 5. $\square$

Lemma 5. $\Pr[\mathsf{A}$ *wins* $\mathsf{Hyb}_5] \leq \frac{(q_h+q)q}{p-1}$.

Proof. Denote $\mathsf{hd}_k \coloneqq \max_{S \subseteq \mathit{Opn}_k, \tilde{\alpha}^{(S)}_k(\mathsf{X}) \neq 0}(\deg(\tilde{\alpha}^{(S)}_k + |S|)$ for each $k \in [q_h]$. Suppose A wins Hyb_5. We have there exists $k \in [q_h]$ such that either there exists $S \subseteq \mathit{Opn}_k$ such that $|S| = 1$ and $\deg(\tilde{\alpha}^{(S)}_k) \geq 1$ or there exists $S \subseteq Opn_k$ such that $|S| \geq 2$ and $\tilde{\alpha}_k^{(S)}(\mathsf{X}) \neq 0$. Therefore, we have $\mathsf{hd}_k \geq 2$. It is not hard to see that the coefficient of $\mathsf{X}^{\mathsf{hd}_k}$ of $\alpha_k(\mathsf{X}) + \alpha'_k(\mathsf{X}) \cdot h$ is

$$\delta_k = \sum_{S \subseteq Opn_k, |S| \leq \mathsf{hd}_k} \tilde{\alpha}^{(S)}_{k, \mathsf{hd}_k - |S|} \prod_{i \in S} e'_i a'_i \ ,$$

where $\tilde{\alpha}^{(S)}_{k,d}$ denotes the coefficient of X^d in $\tilde{\alpha}_k^{(S)}(\mathsf{X})$. Denote

$$Q_k(\{\mathsf{Y}_i\}_{i \in Opn_k}) \coloneqq \sum_{S \subseteq Opn_k, |S| \leq \mathsf{hd}_k} \tilde{\alpha}^{(S)}_{k, \mathsf{hd}_k - |S|} \prod_{i \in S} \mathsf{Y}_i \ .$$

Q_j is a polynomial over $\{\mathsf{Y}_i\}_{i \in Opn_j}$ with degree at most hd_k. Since the abort condition of Hyb_4 does not occur, we have $Q_k(\{\mathsf{Y}_i = e'_i a'_i\}_{i \in Opn_k}) = \delta_k = 0$. Since Q_k is determined when the k-th RO query is made, we know $\{e'_i a'_i\}_{i \in Opn_k}$ are uniformly distributed over $(\mathbb{Z}_p^*)^{|Opn_j|}$ given Q_j. Therefore, by Schwartz-Zippel lemma, we know for any $k \in [q_h]$, $\Pr[Q_k(\{\mathsf{Y}_i = e'_i a'_i\}_{i \in Opn_k}) = 0] \leq \frac{q}{p-1}$. By the union bound, we have $\Pr[\mathsf{A} \text{ wins } \mathsf{Hyb}_5] \leq \Pr[\exists\ k \in [q_h]\ :\ Q_k(\{\mathsf{Y}_i = e'_i a'_i\}_{i \in Opn_k}) = 0] \leq \frac{(q_h+q)q}{p-1}$. □

Remark 1. Our scheme oTZ can be extended for proofs with a *label* attached to the statement, to serve as a message (known only to the user) for a more general blind signature. This would require to slightly change the syntax and accommodate for an additional input $\tau \in \{0,1\}^*$ in the use algorithm, that is $\mathsf{oTZ.Usr}(crs, \boldsymbol{X}, \boldsymbol{Y}, \tau)$ (in free mode) and $\mathsf{oTZ.Usr}(crs, \boldsymbol{X}, \boldsymbol{Y}, \tau)$ (in restricted mode), and compute the challenge from $(\boldsymbol{Y}, Z, \boldsymbol{T}, C, \tau)$. The tag τ would be appended to the final proof. We denote this protocol variant as oTZ_τ.Unforgeability immediately follows from our main theorems (Theorems 3 and 4).

Corollary 1. $\mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ}_\tau,\mathsf{R}}(\lambda) \leq \mathsf{Adv}^{\mathrm{omuf}}_{\mathsf{oTZ},\mathsf{R}}(\lambda)$

Proof. Given an adversary A for the game $\mathrm{OMUF}_{\mathsf{oTZ}_\tau,\mathsf{R},\mathsf{A}}(\lambda)$, we construct an adversary B for the game $\mathrm{OMUF}_{\mathsf{oTZ},\mathsf{R},\mathsf{B}}(\lambda)$. Assume, without loss of generality, that any forgery from A will have an associated random oracle query. Any query to the oracles $\textsc{Issuer}_1$ and $\textsc{Issuer}_2$ is forwarded to the oracles provided by the challenger. Any query to the random oracle (with tag) of the form $\mathrm{H}_\tau(\boldsymbol{Y}, Z, \boldsymbol{T}, C, \tau)$ is dealt lazily as follows: if the input is already present in the hash table H_τ, then just return it; otherwise sample a fresh $\tilde{a}, \tilde{b} \leftarrow\!\!\$\ \mathbb{Z}_p^\times$ and return $\tilde{a}^{-1} \cdot \mathrm{H}(\boldsymbol{Y}, Z, \boldsymbol{T}, \tilde{a}C + \tilde{b}G)$.

Once A returns forgeries $\{(\boldsymbol{Y}_i, Z_i), (a_i, b_i, e_i, \boldsymbol{r}_i, \tau_i)\}_i$, which can be converted into valid forgeries for oTZ: return $\left\{(\boldsymbol{Y}_i, Z_i), (a_i \tilde{a}_{j_i}, \tilde{a}_{j_i} b_i + \tilde{b}_{j_i}, e_i, \boldsymbol{r}_i)\right\}_i$, where $\tilde{a}_{j_i}, \tilde{b}_{j_i}$ denote the randomness used in the random oracle query associated to the i-th forgery. □

Given the above, one-more unforgeability for the blind Schnorr variant given in [43] is immediate. Recall the relation $\mathsf{R}_{\mathsf{sch}}$ that contains all tuples $(x, X, 0, 0) \in \mathbb{Z}_p \times \mathbb{G}^3$, where Y, Z are trivial and always fixed to zero (even when sampled via $\mathsf{SampleArg}$), and also the morphism is trivially $M = [G]$. Then

Corollary 2 ([43]). *The protocol* $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ *for relation* $\mathsf{R}_{\mathsf{sch}}$ *is one-more unforgeable.*

Obliviousness. We show our scheme is oblivious under the discrete logarithm assumption.

Theorem 5. *If DL is hard for* GrGen*, the protocol* $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ *for the DLEQ relation* $\mathsf{R}_{\mathsf{dleq}}$ *is oblivious with advantage*

$$\mathsf{Adv}^{\mathrm{oblv}}_{\mathsf{oTZ}[\mathsf{GrGen},\mathrm{free}],\mathsf{R}_{\mathsf{dleq}}}(\lambda) \leq 2\sqrt{\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda)} + \frac{2}{p} \ ,$$

and the protocol $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ *for any simple algebraic relation* R *is oblivious with advantage*

$$\mathsf{Adv}^{\mathrm{oblv}}_{\mathsf{oTZ}[\mathsf{GrGen},\mathrm{restr}],\mathsf{R}}(\lambda) \leq 2\sqrt{\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda)} + \frac{2}{p} \ .$$

To prove the above theorem, we first consider a special class of adversary, referred to as *argument-honest* adversary, that always sends $Z' = \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$. We show our schemes are perfectly oblivious for any *simple* algebraic relations against any *argument-honest* adversary. Then, for oTZ, we can reduce the obliviousness against any *argument-honest* adversary to the obliviousness against any adversary under the DL assumption.

Proof. (of Theorem 5) We only prove the second half of the statement, i.e., obliviousness of $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$, since the first half is simpler and follows from a similar proof. Let A be an adversary playing the OBLV game for any simple algebraic relation R in restricted mode. Without loss of generality, we assume the randomness of A is fixed. We assume that A always finishes both signing sessions and receives valid proofs (π_0, π_1) from $\textsc{User}_2$. (Otherwise, obliviousness are trivially holds, since the output of $\textsc{User}_0$ and Usr_1 is either Y' and e' in free mode for $\mathsf{R}_{\mathrm{dleq}}$ or *info* and e' in restricted mode, where Y' is uniformly random over $\mathbb{G}^*$ and e' is uniformly random over $\mathbb{Z}_p^*$.)

Denote a bad event Bad as in one of the signing sessions A sends $Z' \neq \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$, where $\boldsymbol{x}$ denotes the discrete logarithm of $\boldsymbol{X}$ and $Y' \leftarrow \mathsf{R.SampleArg}(crs, info)$. We can show that the probability that the bad event occurs is bounded by the advantage of solving the discrete logarithm problem. Suppose the bad event occurs. Since the session does not abort, we obtains a transcript $(\boldsymbol{Y}', Z', \boldsymbol{T}', C', e', r', a', b')$ such that $C' = a'H + b'G$ and $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}' = e'a'[Z'; \boldsymbol{X}] + \boldsymbol{T}'$. By rewinding on the second-round message e', we can obtains another transcript $(\boldsymbol{Y}', Z', \boldsymbol{T}', C', e'', r'', a'', b'')$ such that $C' = a''H + b''G$ and $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}'' = e''a''[Z'; \boldsymbol{X}] + \boldsymbol{T}'$. Since $Z' \neq \langle \boldsymbol{Y}', \boldsymbol{x} \rangle$, it must hold that $e''a'' = a'e'$. Therefore, if $e' \neq e''$, we have $a' \neq a''$ and thus we can extract the discrete logarithm to the base G of H as $(b'' - b')/(a' - a'')$. By the forking lemma, we have $\Pr[\mathsf{Bad}] \leq 2\sqrt{\mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen}}(\lambda)} + \frac{2}{p}$.

We now show that the protocol is perfectly oblivious given the bad event does not occur.

Let V_A denote the set of all possible views of A that can occur after finishing both signing sessions. In particular, any such view $\Delta \in V_A$ takes form $\Delta = (\boldsymbol{X}, \boldsymbol{Y}_0, Z_0, \boldsymbol{Y}_1, Z_1, \tau_0, \tau_1, \pi_0, \pi_1)$. (We can ignore *info* for restricted mode since it is fixed given A is fixed.) Here, $\pi_i = (a_i, b_i, \boldsymbol{r}_i, e_i)$, where $e_i = \mathrm{H}(\boldsymbol{Y}_i, Z_i, [\boldsymbol{Y}_i; M]\boldsymbol{r}_i, a_iH + b_iG)$. Moreover, τ_0 and τ_1 are the issuing protocol transcripts for session 0 and 1, respectively, and take form

$$\tau_i = (\boldsymbol{Y}_i', Z_i', \boldsymbol{T}_i', C_i', e_i', \boldsymbol{r}_i', a_i', b_i') .$$

We need to show that the distribution of the actual adversarial view, which we denote as v_A, is the same when $b = 0$ and $b = 1$. Because we assume the randomness of A is fixed, the distribution of v_A only depends on the randomness $\eta = (\upsilon_0, \varepsilon_0, \alpha_0, \beta_0, \boldsymbol{\rho}_0, \upsilon_1, \varepsilon_1, \alpha_1, \beta_1, \boldsymbol{\rho}_1)$ required to respond to $\textsc{User}_0$, $\textsc{User}_1$ and $\textsc{User}_2$ queries, and we write $v_A(\eta)$ to make this fact explicit.

Concretely, fix some $\Delta \in V_A$. We now show that there exists a unique η that makes it occur, i.e, $v_A(\eta) = \Delta$, regardless of whether we are in the $b = 0$ or in the $b = 1$ case. In particular, we claim that, in both cases $b = 0, b = 1$, $v_A(\eta) = \Delta$ if and only if for $i \in \{0, 1\}$, η satisfies

$$\begin{aligned} \boldsymbol{Y}'_{\omega_i} &= \upsilon_i \boldsymbol{Y}_i \\ \alpha_i &= a'_{\omega_i}/a_i \\ \beta_i &= \alpha_i^{-1} b'_{\omega_i} - b_i \\ \varepsilon_i &= \alpha_i e'_{\omega_i} e_i \\ \boldsymbol{\rho}_i &= \boldsymbol{r}_{\omega_i} - \varepsilon_i \boldsymbol{r}_i \end{aligned} \tag{8}$$

where $\omega_0 = b$ and $\omega_1 = 1 - b$. It is hard to see that there exists unique (υ_0, υ_1) and thus a unique η satisfies Equation (8). In free mode for R_{dleq}, υ_i must be equal to $\log_G Y'_{\omega_i} / \log_G Y_i$. In restricted mode, since A is argument-honest, such υ_i must exist and both Y_0' and Y_1' are not 0, which implies the uniqueness of υ_i.

To prove the above claim, in the "only if" direction, from Figures 6 and 7, it is clear that when $v_A(\eta) = \Delta$, then η satisfies all constraints in Equation (8).

To prove the "if" direction, assume that η satisfies all constraints in Equation (8). We need to show that $v_A(\eta) = \Delta$. This means in particular verifying that in free mode, $\textsc{User}_0$ indeed outputs Y_0' and Y_1', and in both modes, the challenges output by $\textsc{User}_1$ and the proofs output by $\textsc{User}_2$ are indeed $(e_0, (Y_0, Z_0), \pi_0)$ and $(e_1, (Y_1, Z_1), \pi_1)$. It is clear that the output $Y_0', Y_0, Z_0, Y_1', Y_1, Z_1$ are consistent with Δ.

For the challenges, note that because we only consider Δ's that result in Usr_2 not producing output $(\perp, \perp)$, we have

$$e_i = \mathrm{H}(\boldsymbol{Y}_i, Z_i, [\boldsymbol{Y}_i; M] \cdot \boldsymbol{r}_i - e_i a_i [Z_i; \boldsymbol{X}], a_iH + b_iG) .$$

$$
\begin{array}{lcl}
\underline{\mathsf{oVuf.Usr}(\mathit{crs}, \mathit{pk}, m)} & & \underline{\mathsf{oVuf.Iss}(\mathit{crs}, \mathit{sk})} \\
Y := \mathrm{H}_1(m) & & \\
\mathit{st}, Y' \leftarrow \mathsf{oTZ.Usr}_0(\mathit{crs}, \mathit{pk}, Y) & & \\
 & \xrightarrow{\quad Y' \quad} & \\
 & \xleftarrow{\quad Z', \boldsymbol{T}', C' \quad} & \mathit{st}, (Z', \boldsymbol{T}', C') \leftarrow \mathsf{oTZ.Iss}_1(\mathit{crs}, \mathit{sk}, \mathit{pk}, Y') \\
e' \leftarrow \mathsf{oTZ.Usr}_1(\mathit{st}, (Z', \boldsymbol{T}', C')) & \xrightarrow{\quad e' \quad} & \\
 & \xleftarrow{\quad r', a', b' \quad} & (r', a', b') \leftarrow \mathsf{oTZ.Iss}_2(\mathit{st}, e') \\
\textbf{return } (Y, Z), \pi := (a, b, e, r) \leftarrow \mathsf{oTZ.Usr}_2(\mathit{st}, (r', a', b')) & &
\end{array}
$$

Fig. 9. The issuance protocol for the OVUF protocol oVuf from Section 5.2.

Since

$$
\begin{aligned}
T_i &= \varepsilon_i^{-1} \begin{pmatrix} \upsilon_i^{-1} & 0 \\ 0 & I_n \end{pmatrix} \left(\boldsymbol{T}'_{\omega_i} - M_{\boldsymbol{Y}'_{\omega_i}} \boldsymbol{\rho}_i \right) \\
&= \varepsilon_i^{-1} \begin{pmatrix} \upsilon_i^{-1} & 0 \\ 0 & I_n \end{pmatrix} \left([\boldsymbol{Y}'_{\omega_i}; M] \cdot \boldsymbol{r}'_{\omega_i} - e'_{\omega_i} a'_{\omega_i} [Z'_{\omega_i}; \boldsymbol{X}] - [\boldsymbol{Y}'_{\omega_i}; M] \boldsymbol{\rho}_i \right) \\
&= [\boldsymbol{Y}_i; M] \cdot \varepsilon_i^{-1} (\boldsymbol{r}_i - \boldsymbol{\rho}_i) - \varepsilon_i^{-1} e'_{\omega_i} a'_{\omega_i} [Z_i; \boldsymbol{X}] \ , \\
&= [\boldsymbol{Y}_i; M] \cdot \boldsymbol{r}_i - e_i a_i [Z_i; \boldsymbol{X}] \ ,
\end{aligned}
$$

$$
\begin{aligned}
C_i &= \alpha_i^{-1} C'_{\omega_i} - \beta_i G \\
&= \alpha_i^{-1} (a'_{\omega_i} H + b'_{\omega_i} G) - \beta_i G \\
&= \alpha_i^{-1} a'_{\omega_i} H + (\alpha_i^{-1} b'_{\omega_i} - \beta_i) G \\
&= a_i H + b_i G \ ,
\end{aligned}
$$

the challenge output by User_1 are indeed e'_i. Then, by Equation (8), it is clear that the output proof are indeed π_i. $\square$

5 Oblivious Verifiable Unpredictable Functions

We rely on oblivious zero-knowledge proofs to build an *Oblivious Verifiable Unpredictable Function* (OVUF) from pairing-free groups of prime order. As in a verifiable unpredictable function (VUF), a weakening of a VRF [33], we consider a setting where an issuer holds a secret key sk, the user knows a public key pk, and they engage in an interactive protocol to jointly evaluate a function $Z = \mathsf{F}(\mathit{sk}, m)$ of an input m chosen by the user. The user learns Z, along with a proof π that attests that $Z = \mathsf{F}(\mathit{sk}, m)$, which is verified with help of the public key. Crucially, however, we require this evaluation to be oblivious–the issuer does not learn anything about m, Z, and π during the execution. We note that this notion is stronger than that of a (verifiable) OPRF, in that the latter only provides verifiability to the user, as the issuer provides a linkable proof of evaluation which cannot be made public.

Before we turn to the formal treatment of OVUFs, and our construction, we observe that an OVUF directly yields a blind signature scheme producing signatures $\sigma = (\mathsf{F}(\mathit{sk}, m), \pi)$ for a message m–therefore, the first part of the signature is *unique* in that it only depends on m and sk. This is a natural weakening of the notion of unique signatures [26], which suffices in many of their applications. To the best of our knowledge, no unique signatures, or partially unique ones, are known in the pairing-free setting, let alone blind ones. This is contrast to the pairings setting, where BLS signatures [8] and their blind version [7] are unique. We expand on this further below.

5.1 Syntax and security

An OVUF protocol consists of a tuple of p.p.t. algorithms $\mathsf{oVuf} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Iss}, \mathsf{Usr}, \mathsf{F}, \mathsf{Ver})$, with the following functionalities:

- $\mathit{crs} \leftarrow \mathsf{oVuf.Setup}(1^\lambda)$, the setup algorithm, generates the public parameterscrs
- $(\mathit{sk}, \mathit{pk}) \leftarrow \mathsf{oVuf.KeyGen}(\mathit{crs})$, the key generation algorithm, generates a *secret key sk* and a *public verification key pk*
- The interactive algorithms $\mathsf{oVuf.Iss}$ (the *issuing* algorithm) and $\mathsf{oVuf.Usr}$ (the *user* algorithm) take as input $(\mathit{sk}, \mathit{pk})$, and (pk, m), respectively, along with crs. The interaction and the outputs of the issuer and user are denoted as:

$$((Z, \pi), \bot) \leftarrow \langle \mathsf{oVuf.Usr}(\mathit{crs}, \mathit{pk}, m), \mathsf{oVuf.Iss}(\mathit{crs}, \mathit{sk}, \mathit{pk})\rangle$$

  where $m \in \{0,1\}^*$ is a string. Moreover, we require that $Z = \mathsf{F}(\mathit{crs}, \mathit{sk}, m)$ is the *unique* output of the associated key function F.
- $\mathbf{true}/\mathbf{false} \leftarrow \mathsf{oVuf.Ver}(\mathit{crs}, \mathit{pk}, m, Z, \pi)$ outputs a bit to indicate whether π is a valid proof that $Z = \mathsf{F}(\mathit{crs}, \mathit{sk}, m)$.

We require a number of security properties for an OVUF, which we state here only informally. (A proper formalization follows along the lines of Section 3.)

- *Soundness.* Any p.p.t. adversary playing the role of a malicious user, given crs and pk, should not be able to interact with an honest issuer (in an a-priori unbounded polynomial number of concurrent executions) and generate a triple (m^*, Z^*, π^*) such that $\mathsf{oVuf.Ver}(\mathit{crs}, \mathit{pk}, m^*, Z^*, \pi^*)$ is $\mathbf{true}$, but $\mathsf{F}(\mathit{crs}, \mathit{sk}^*, m^*) \neq Z^*$.
- *One-more unforgeability.* The security game initially runs $\mathsf{oVuf.Setup}(1^\lambda)$ to generate crs and $(\mathit{sk}, \mathit{pk}) \leftarrow \mathsf{oVuf.KeyGen}(\mathit{crs})$. The p.p.t. adversary, given crs and pk, can then interact concurrently over ℓ sessions with $\mathsf{oVuf.Iss}(\mathit{crs}, \mathit{sk})$. It wins if it outputs $\ell + 1$ distinct triples $\{(m_j, Z_j, \pi_j)\}_{j \in [\ell+1]}$ such that $\mathsf{oVuf.Ver}(\mathit{crs}, \mathit{pk}, m_j, Z_j, \pi_j) = \mathbf{true}$ for all $j \in [\ell + 1]$.
- *One-more unpredictability.* The security game initially runs $\mathsf{oVuf.Setup}(1^\lambda)$ to generate crs and $(\mathit{sk}, \mathit{pk}) \leftarrow \mathsf{oVuf.KeyGen}(\mathit{crs})$. The adversary, given crs and pk, interacts with $\mathsf{oVuf.Iss}(\mathit{crs}, \mathit{sk})$ in ℓ concurrent sessions. It wins if it outputs $\ell+1$ distinct pairs $\{(m_i, Z_i)\}_{i \in [\ell+1]}$ such that $\mathsf{F}(\mathit{crs}, \mathit{sk}, m_i) = Z_i$ for all $i \in [\ell + 1]$.
- *Obliviousness.* We can define obliviousness with respect to a cheating issuer in a way very similar to that of what done in Section 3, which guarantees that any triple (m, Z, π) output by an honest user cannot be linked back to which issuance session that generates it. We omit the formal definition here.

One-more unpredictability is a natural relaxation (to the oblivious setting) of unpredictability for VUFs. It captures the fact that only ℓ evaluations of $\mathsf{F}(\mathit{sk}, \cdot)$ are learnt through ℓ interactions with the issuer. It is not implied by one-more unforgeability, as it may be easier to break it if we are not asked to *also* generate a proof. The converse is not true either since we may be able to break one-more unforgeability by presenting $\ell + 1$ proofs for a single pair (m, Z).

5.2 An OVUF protocol and its security

An OVUF protocol $\mathsf{oVuf}[\mathsf{GrGen}]$ is easily obtained from $\mathsf{oTZ} = \mathsf{oTZ}[\mathsf{GrGen}, \mathrm{free}]$ for the DLEQ relation $\mathsf{R}_{\mathrm{dleq}}$. We give it for completeness:

$$\mathsf{R}_{\mathrm{dleq}} := \left\{ (x, X, Y, Z) \ : \ \forall\, x \in \mathbb{Z}_p \text{ and } Y \in \mathbb{G}, \quad \begin{bmatrix} Z \\ X \end{bmatrix} = \begin{bmatrix} Y \\ G \end{bmatrix} \cdot x \right\} .$$

The relation-specific setup is empty, i.e.: $\mathsf{R}_{\mathrm{dleq}}.\mathsf{Setup}(\Gamma) = \bot$.

- The Setup algorithm, on input 1^λ, runs $(\mathbb{G}, p, G) \leftarrow\!\!\$\ \mathsf{GrGen}(1^\lambda)$, and samples a second generator $H \leftarrow\!\!\$\ \mathbb{G}$. It also implicitly defines two hash functions $\mathrm{H}_1 : \{0,1\}^* \to \mathbb{G}$ and $\mathrm{H} : \{0,1\}^* \to \mathbb{Z}_p^*$. Finally, it returns $\mathit{crs} := (\mathbb{G}, p, G, H)$. ($\Lambda$ is empty.) We stress here that the particular choice of the hash functions is not spelled out further, as we will assume them to be random oracles in our security analysis.
- The KeyGen algorithm, on input $\mathit{crs} = (\mathbb{G}, p, G, H)$, picks $\mathit{sk} \leftarrow \mathbb{Z}_p$ and sets $\mathit{pk} := \mathit{sk} \cdot G$. It returns $(\mathit{sk}, \mathit{pk})$.

- We define $\mathsf{F}(crs, sk, x) = sk \cdot \mathrm{H}_1(x)$.
- The Iss and Usr algorithms are derived from those of $\mathsf{oTZ} = \mathsf{oTZ}[\mathsf{GrGen}, \text{free}]$. (Note in particular that (sk, pk) are a sample from $\mathsf{Core}(\mathsf{R}_{\text{dleq}})$.) In particular,

$$\begin{aligned} \mathsf{Usr}(crs, pk, m) &= \mathsf{oTZ}.\mathsf{Usr}(crs, pk, \mathrm{H}_1(m)) \ , \\ \mathsf{Iss}(crs, sk) &= \mathsf{oTZ}.\mathsf{Iss}(crs, sk, pk = sk \cdot G) \ . \end{aligned} \tag{9}$$

  The original user algorithm $\mathsf{oTZ}.\mathsf{Usr}$ would return a tuple $(Y, Z, \pi := (a, b, e, r))$, but $Y = \mathrm{H}_1(m)$ is redundant here, and therefore only $(Z, \pi := (a, b, e, r))$ is returned by $\mathsf{Usr}(crs, pk, m)$. Clearly, in an honest execution, $Z = \mathsf{F}(crs, sk, m)$. We provide a self contained description of the protocol in **??**.
- The verification algorithm, given m, Z, and $\pi = (a, b, e, r)$, first computes $Y := \mathrm{H}_1(m)$, and then verifies that π is a valid proof for $(sk, pk, Y, Z) \in \mathsf{R}_{\text{dleq}}$. This is done by computing

$$C := aH + bG \ ,$$

  as well as $\boldsymbol{T} := r[Y; G] - ea[Z; pk]$. Then, we check that $\mathrm{H}(Y, Z, \boldsymbol{T}, C) = e$.

Security. It is not hard to see that the obliviousness of oVuf is implied the obliviousness of oTZ. We show other security guarantees in the following lemmas.

Lemma 6. oVuf *achieves soundness if* H *is a random oracle.*

Proof (Proof Sketch). By the soundness of oTZ, it follows by standard techniques that an unbounded adversary, on input crs and $pk = sk \cdot G$, querying H a polynomial number of times, cannot output $(m, Z, \pi = (a, b, e, r))$ such that π is valid and $(sk, pk, \mathrm{H}_1(m), Z) \notin \mathsf{R}_{\text{dleq}}$. (For this argument, the hash function H_1 can be fixed, and does not need to be a random oracle.) Clearly for such a prover access to the issuer does not help, as the unbounded prover knows sk without loss of generality, and can simulate the issuer on its own.

Lemma 7. oVuf *is one-more unforgeable under the* $(q+1)$*-SDL assumption in the algebraic group model and in the random oracle model.*

Proof (Proof sketch.). Here, we assume that both H and H_1 are random oracles. The high-level idea is simple: A winning adversary A against one-more unforgeability of oVuf, on input crs, pk, would output $\{(m_j^*, Z_j^*, \pi_j^*)\}_{j \in [\ell+1]}$ such that $\mathsf{oVuf}.\mathsf{Ver}(crs, m_j^*, Z_j^*, \pi_j^*) = \textbf{true}$ for all $j \in [\ell + 1]$. This yields an adversary B against OMUF security of $\mathsf{oTZ}[\mathsf{GrGen}, \text{free}]$, which, on input $X = xG$ and crs, runs A on input crs and X, simulates the issuer of oVuf with its oracles. Finally, it outputs $\{(\mathrm{H}_1(m_j^*), Z_j^*, \pi_j^*)\}_{j \in [\ell+1]}$. Clearly, B wins if A wins, or if a collision for H_1 was found.

To use Theorem 3, however, we need to make sure that B is an algebraic adversary of the right format. The problem is that A can supply algebraic representations of elements that also depend on the outputs of H_1–as B has no access to a second oracle H_1, such representations cannot be output by B. This is easy to overcome, however, by letting B simulate H_1 to A so that the discrete logarithm h_m of the result $\mathrm{H}_1(m) = h_m G$ of each query m is known to B. This then allows B to convert all representations supplied by A in terms of the group elements input to B only.

Lemma 8. oVuf *is one-more unpredictable under the one-more gap-DH assumption in the random oracle model.*

Here, we do not state explicitly the one-more gap DH assumption, but note that this is the exact assumption needed to prove one-more unpredictability in the random oracle model in a setting where the issuer merely provides $sk \cdot Y'$ on input Y', without generating a proof. (This follows from [7].) The proof of the lemma then uses the specific structure of our proof to simulate the proof issuance without knowledge of sk by ensuring that the reduction generates the generator H with a known discrete log.

Proof (Proof sketch). It is well known that if an adversary B is only given access to an issuer which returns $Z' := skY'$ on input Y', then one-more unpredictability holds under the one-more gap-DH assumption in the random oracle model [7]. However, in our setting, the adversary A interacts with the full issuer, which also engages into the issuance of a proof π, and we need to prove that it does not help. Here, we argue that the adversary B can run A and simulate the full issuance by setting up parameters such that the discrete logarithm $H = \alpha G$ of the second generator H is known to B. In particular, when A starts an interaction with the issuer with value Y', B call its oracle to obtain $Z' := sk \cdot Y'$. Then, it also generates

$$\eta, t', r', c^* \leftarrow \mathbb{Z}_p \;, \quad C' := \eta \cdot G \;, \;\; \boldsymbol{T}' := r' \begin{bmatrix} Y' \\ G \end{bmatrix} - c^* \begin{bmatrix} Z' \\ pk \end{bmatrix} \;, .$$

It returns $Z', \boldsymbol{T}', C'$ to A. If A then continues this session with a challenge e', B computes $a' := e' \cdot (c^*)^{-1}$. It also compute b' such that $\alpha a' + b' = \eta$, and returns r', a', b' to A.

Remark 2. Given $sk \cdot H_1(m)$ is a pseudorandom function in the ROM, one may wonder whether we could obtain the stronger object of an oblivious verifiable random function (OVRF), instead. However, we note that existing proofs for the weaker notion of VOPRF security [30,44] all extend the above PRF with an outer hash, which would interfere with our ability of generating efficient proofs.

5.3 Applications

Partially unique blind signatures. We can think of the above OVUF protocol as a blind signature scheme, with signatures of form $\sigma = (sk \cdot \mathrm{H}(m), \pi)$, where π is a proof that ensures correctness of the first portion of the signature, which is verified with the public key $pk = sk \cdot G$. One-more unforgeability directly implies one-more unforgeability of the blind signature. Further, soundness guarantees that for any valid signature we indeed have $Z = sk \cdot \mathrm{H}_1(m)$, i.e., the Z part is a deterministic function of m and sk. Such signatures (as in unique signatures) can save verification costs by avoiding verifying multiple signatures for the same message over and over. Once $\sigma = (sk \cdot \mathrm{H}(m), \pi)$ is verified, every following signature on m will also contain the same $sk \cdot \mathrm{H}(m)$ portion.

From a theoretical perspective, the resulting scheme offers an alternative to existing blind signature schemes (e.g. [1,29,24,43]) in pairing-free groups. While we rely on four messages, as opposed to three in these schemes, this difference is immaterial as three-message schemes also require two round trips.

Hybrid publicly/privately verifiable tokens. Our construction also gives new approaches to *anonymous tokens*, as e.g. in PrivacyPass [19,13]. The original protocol [19] relies on *privately* verifiable tokens functionally equivalent to $(r, sk \cdot \mathrm{H}_1(r))$ for a random r. (The 2HashDH OPRF [30] was used instead, but it is easy to see that pseudorandomness is not needed.) However, many situations call for *publicly* verifiable tokens, and in practice, PrivacyPass usually relies on a blind signature on r instead.

Our solution offers a hybrid approach, where the token $(r, sk \cdot \mathrm{H}_1(r))$ is issued with an unlinkable publicly verifiable *proof* π. If the token verifier knows the secret key, it can use it to very quickly verify $(r, sk\mathrm{H}_1(r))$. Unpredictability still guarantees that these (privately-verifiable) tokens are secure. However, third parties that only know the public key can also, less efficiently, verify the token with the help of π.

6 Public verification for algebraic MACs

Keyed-verification anonymous credentials (KVACs) [15] are typically constructed with an algebraic MAC, (this is a direct analog of anonymous credentials constructed from blind signatures). In this approach, the issuer creates a MAC σ on a list of attributes $\boldsymbol{m}$, sends σ to the user. Since the user is not able to verify σ (unlike the blind signature case), the issuer also provides an *issuance proof* π that proves σ is well-formed with respect to a commitment to the MAC key and $\boldsymbol{m}$. This is important for unlinkability of the credential, since otherwise σ may be invalid in a way that is unique to the issuance session.

Here we apply our transform to blind π, effectively making it a blind signature on $\boldsymbol{m}$. This is interesting for three reasons. First, since π can be verified by anyone, we can upgrade a KVAC to have some of the public verifiability present in traditional anonymous credential systems, while still retaining the very efficient KVAC protocols for cases when public verifiability is not required. Second, there are many KVAC constructions (based on different algebraic MACs) we can potentially upgrade in this way, with different features, tradeoffs and assumptions. Third, this construction provides some of the functionality of anonymous credentials with concurrent security.

Technical Overview. We use a specific, efficient MAC, called $\mathsf{MAC}_{\mathrm{GGM}}$ in [15], to describe our approach but we believe that a similar reasoning applies also to $\mathsf{MAC}_{\mathrm{DDH}}$ (another MAC from [15]). We describe the case of a single attribute. The setup algorithm generates the public parameters $(\Gamma = (\mathbb{G}, p, G), W)$ where W is a generator of $\mathbb{G}$ whose discrete logarithm w.r.t. G is not known. The secret key is $\boldsymbol{x} \coloneqq [x_{-1}; x_0; x_1]$ and the public parameters are $(X_0, X_1) = (x_0 G + x_{-1} W, x_1 W)$. To compute a MAC on message m, sample Y' at random in $\mathbb{G}$, then output $\sigma = (Y', Z') = (Y', (x_0 + x_1 m)Y')$. To verify (σ, m), use Y' and (x_0, x_1) to recompute Z'' and accept if $Z'' = Z'$. The proof π is proof of knowledge of (x_{-1}, x_0, x_1) such that

$$Z' = (x_0 + x_1 m)Y' \;\wedge\; X_1 = x_1 W \;\wedge\; X_1 = x_0 G + x_{-1} W$$

which can be realized with a generalized Schnorr proof, amenable to our transform. We also note that a MAC (Y', Z') can be re-randomized to a MAC $(Y, Z) = (\upsilon Y', \upsilon Z')$ where υ is a random value.

6.1 Syntax

Syntax for Algebraic MACs. A generic algebraic MAC with public issuance, denoted MAC, has the following algorithms. Denote with $\mathsf{MAC.Setup}(\mathit{crs})$ be the setup algorithm that, given as input the security parameter in unary form 1^λ, outputs some common reference string crs. $\mathsf{MAC.KeyGen}(\mathit{crs})$ generates a key pair $(\mathit{sk}, \mathit{pk})$ where sk is the secret key, and pk are the public parameters (used only when proving statements about an authentication tag). $\mathsf{MAC.MAC}(\mathit{crs}, \mathit{sk}, \boldsymbol{m})$ outputs an authentication tag μ, on input message $\boldsymbol{m}$ (which we allow to be a vector of messages, also called attributes). $\mathsf{MAC.sVer}(\mathit{crs}, \mathit{sk}, \boldsymbol{m}, \mu)$ verifies that μ is a valid authentication tag for $\boldsymbol{m}$.

Syntax for Publicly Verifiable Algebraic MACs. The setup algorithm is run by a trusted party and outputs crs, describing a group description Γ for a group $\mathbb{G}$ of order p and three non-trivial generators G, H (the commitment key for Ped), and W (the crs of MAC) . The signer uses $\mathsf{KeyGen}(\mathit{crs})$ to create a keypair $(\mathit{sk}, \mathit{pk})$. Signature generation (or issuance) is a three-message protocol between the User and Issuer. In its basic instantiation, both parties share pk, and the signer also knows sk. The user shares the message vector $\boldsymbol{m}$ (denoted *info* in the generic protocol) with the issuer. [8] The output is a first message Imsg_1, sent from issuer to user, and we write $(\mathit{Imsg}_1, \mathit{st}) \leftarrow \mathsf{MAC.Sign}_1(\mathit{crs}, \mathit{sk}, \mathit{pk}, \boldsymbol{m})$ (where st is state required by the Issuer for the rest of the protocol) to denote the first message produced by the issuer. In the second step the user has input Imsg_1, sends a message Umsg to the issuer via the user algorithm $\mathsf{MAC.Usr}_1$, which in turn creates a response, denoted $\mathit{Imsg}_2 \leftarrow \mathsf{MAC.Sign}_2(\mathit{st}, e')$. Then the User locally computes the proof π, from $(\mathit{Imsg}_1, \mathit{Imsg}_2)$ and their state. The signature is verified with $\mathsf{MAC.Ver}(\mathit{crs}, \mathit{pk}, \boldsymbol{m}, \pi)$. (We stress that $\mathsf{oCMZ.sVer}$ denotes secret-key verification, while $\mathsf{oCMZ.Ver}$ denotes public-key verification.) In our constructions we have the property that from π it is always possible to parse out an algebraic MAC μ on the same message. This is necessary to allow the dual-use feature of the credential (both as a KVAC and a blind signature).

We informally define security of publicly verifiable algebraic MACs with the following two properties:

– *Unlinkability.* The challenger sets up the public parameters crs and internally runs the p.p.t. adversary $\mathsf{A}(\mathit{crs})$ and returns a public-key pk, with a message vector $\boldsymbol{m}$. The challenger samples $b \leftarrow\!\!\$\; \{0,1\}$, and lets the adversary interact with two user session $\mathsf{Usr}(\mathit{pk}, \boldsymbol{m})$, with the same $\boldsymbol{m}$, potentially interleaving messages across different rounds. Finally, if neither instances failed, A gets the resulting proofs in a random order $\pi^{(b)}, \pi^{(1-b)}$ and outputs a guess $b' \in \{0,1\}$. If $b = b'$, the adversary wins the game. The adversary has also at disposal a verification oracle $\textsc{Verify}$ that internally runs $\mathsf{MAC.sVer}$ for secret-key verification.

– *Unforgeability* for n attributes. Here, the adversary A engages in polynomially many (in λ) adaptive interactive protocols with the issuer (implemented by the challenger). The challenger and A proceed for ℓ issuance sessions, for arbitrary messages chosen by the adversary. At the end of its execution, A outputs a set of proofs and corresponding attributes $\{(\pi_j, \boldsymbol{m}_j)\}_j$, and wins the game if:
  (a) *(one-more forgery)* A outputs at least $\ell + 1$ proofs that are all valid and different, or
  (b) *(attribute forgery)* one of proofs output by A is valid and has an associated attribute vector $\boldsymbol{m}$ that was not queried during issuance.

[8] This is also a simplification; in Section 6.3 we describe how $\boldsymbol{m}$ may be (partially) hidden during issuance (Section 7.3).

$$\begin{array}{lcl}
\underline{\mathsf{oCMZ.Usr}(crs, \boldsymbol{X}, \boldsymbol{m})} & & \underline{\mathsf{oCMZ.Iss}(crs, \boldsymbol{x}, \boldsymbol{X})} \\
& \xrightarrow{\qquad \boldsymbol{m} \qquad} & \\
& \xleftarrow{\qquad Z', \boldsymbol{T}', C' \qquad} & st, (Z', \boldsymbol{T}', C') \leftarrow \mathsf{oTZ.Iss}_1(crs, \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{m}) \\
st, e' \leftarrow \mathsf{oTZ.Usr}_1(crs, \boldsymbol{X}, \boldsymbol{m}, (Z', \boldsymbol{T}', C')) & \xrightarrow{\qquad e' \qquad} & \\
& \xleftarrow{\qquad r', a', b' \qquad} & r', a', b' \leftarrow \mathsf{oTZ.Iss}_2(st, e') \\
\textbf{return } ((Y,Z),(a,b,e,r)) \leftarrow \mathsf{oTZ.Usr}_2(st,(r',a',b')) & &
\end{array}$$

Fig. 10. Oblivious issuance protocol for CMZ credentials.

Both security properties are inspired from the notion of *blind signatures with attributes* in [3, Def. 6 and Def. 7]. When extended to handle private attributes (cf. Section 6.3), the differences are in: (1) the syntax for the algorithms, which is slightly different than [3]; (2) in unlinkability, where we do not require the key pair to be honestly generated; (3) in the unlinkability property, we clarify that unlinkability can hold only for attributes whose public information is the same.

6.2 Oblivious issuance proof

We now define the new issuance proof for CMZ credentials, denoted oCMZ. As the original protocol, it provides a proof π attesting the validity of the given credential, but now π is created with our oblivious ZK proof framework. The procedure $\mathsf{oCMZ.Setup}(1^\lambda)$ invokes the setup for the relation $\mathsf{R}_{\mathrm{MAC}}.\mathsf{Setup}(\Gamma)$ which samples and returns the generator $W \in \mathbb{G}$ such that its DL w.r.t. G and H is not known. In other words, $crs = (\Gamma, H, W) \leftarrow \mathsf{oCMZ.Setup}(1^\lambda)$. The key generation algorithm $\mathsf{oCMZ.KeyGen}(crs)$ outputs a new key pair (sk, pk), where the secret key is $sk := \boldsymbol{x} := [x_{-1}; x_0; x_1; \ldots; x_n]$ chosen at random and the public key is $pk := \boldsymbol{X} \in \mathbb{G}^{n+1}$ is constructed as $X_0 := x_0 G + x_{-1} W$, $X_i = x_i W$ for $1 < i \leq n$. A MAC on a (public) vector $\boldsymbol{m}$ is generated via the procedure $\mathsf{oCMZ.Sign}$ that internally runs the generic restricted protocol $\mathsf{oTZ}[\mathsf{GrGen}]$ (illustrated in Figure 6, and re-stated in Figure 10) by setting $info := \boldsymbol{m} \in \mathbb{Z}_p^n$ to be the message vector to be signed $\boldsymbol{m}$ and the relation to be proven as

$$\begin{aligned}\mathsf{R}_{\mathrm{CMZ}} := \{(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y} = (0, Y, \boldsymbol{m} \cdot Y), Z) \in \mathbb{Z}_p^{n+2} \times \mathbb{G}^{n+1} \times \mathbb{G}^{n+2} \times \mathbb{G} :\ & [\boldsymbol{Y}; M] \cdot \boldsymbol{x} = [Z; X_0; \ldots X_n] \textbf{ and} \\ & X_0 = x_0 G + x_{-1} W \ \textbf{ and } \ X_i = x_i W \text{ for } i \in [1, n]\}\end{aligned}$$

where the morphism is

$$\begin{bmatrix} \boldsymbol{Y} \\ M \end{bmatrix} := \begin{bmatrix} 0 & Y & m_1 Y & m_2 Y & \cdots & m_n Y \\ W & G & 0 & 0 & \cdots & 0 \\ 0 & 0 & W & 0 & \cdots & 0 \\ 0 & 0 & 0 & W & \cdots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & W \end{bmatrix}. \tag{10}$$

The setup algorithm $\mathsf{R}_{\mathrm{CMZ}}.\mathsf{Setup}(\Gamma)$ samples W uniformly from $\mathbb{G}^*$ and returns $\Lambda = (\Gamma, W)$. The argument sampling procedure $\mathsf{R}_{\mathrm{CMZ}}.\mathsf{SampleArg}(crs, \boldsymbol{m})$ returns a tuple $\boldsymbol{Y} := (0, Y, \boldsymbol{m} \cdot Y) = (0, Y, m_1 Y, m_2 Y, \ldots, m_n Y)$ for some Y randomly sampled from $\mathbb{G}^*$. However, we assume $\mathsf{oCMZ.Sign}$ does not send the whole vector $\boldsymbol{Y}$, and instead simply sends Y as the other elements can be reconstructed from the message vector $\boldsymbol{m}$.

It is clear that $\mathsf{R}_{\mathrm{CMZ}}$ is a simple algebraic relation (cf. Definition 1), since one generates M and $\boldsymbol{Y}$ with their DLs to base G and fixed $(\Lambda, \boldsymbol{m})$, any $\boldsymbol{Y}_1, \boldsymbol{Y}_2 \in \mathsf{Arg}(\mathsf{R}_{\mathrm{CMZ}})$ is such that $\exists r \in \mathbb{Z}_p$ such that $rY_1 = Y_2$. Also, the kernel Diffie-Hellman problem is hard for $\mathsf{R}_{\mathrm{CMZ}}$, since finding non-zero $\boldsymbol{r} \in \mathbb{Z}_p^{n+2}$ such that $M\boldsymbol{r} = 0$ implies solving the DL of W to base G.

At the end of the issuance protocol, the user has a MAC $\mu := (Y, Z)$ and an unlinkable proof $\pi := (a, b, e, \boldsymbol{r})$ on $\boldsymbol{m}$. The public verification algorithm $\mathsf{oCMZ.Ver}$ takes as input $\boldsymbol{X}$, the message $\boldsymbol{m}$,

```
$\mathsf{oCMZ.Ver}(crs, \boldsymbol{X}, \boldsymbol{m}, (Y, Z), (a, b, e, \boldsymbol{r}))$
$\boldsymbol{Y} := (0, Y, m_1 Y, m_2 Y, \dots, m_n Y)$
// Defined in Figure 5
return $\mathsf{oTZ.Ver}(crs, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, (a, b, e, \boldsymbol{r}))$
```

```
$\mathsf{oCMZ.sVer}(crs, \boldsymbol{x}, \boldsymbol{m}, (Y, Z))$
// $\mathsf{MAC_{GGM}.Ver}$ from [15]
return $\left((\sum_i x_i m_i + x_0)Y = Z\right)$
```

Fig. 11. The public and secret-key verification algorithm for our MAC-based anonymous credentials.

together with the MAC (Y, Z) and the proof π. It derives $\boldsymbol{Y} = (0, Y, \boldsymbol{m} \cdot Y)$ and internally invokes our generic verifier in restricted mode (Figure 5). The following lemma follows from correctness and obliviousness (cf. Theorem 5) of the generic protocol $\mathsf{oTZ}[\mathsf{GrGen}]$.

Lemma 9. *The algebraic MAC with public verification* oCMZ *is correct and unlinkable.*

Unforgeability. We prove security of our construction in two parts: we first study (pure) one-more unforgeability, corresponding to the winning event (a), and then attribute unforgeability, corresponding to the winning event (b) on Page 24. The case (a) follows from the main theorem (cf. Theorem 3), while the case (b) is more involved and requires a reduction to the standard unforgeability of algebraic MACs.

Lemma 10. *If* $\mathsf{MAC_{GGM}}$ *is* $\mathsf{uf\text{-}cmva}$ *secure [15, Def. 1] and DL is hard for* GrGen*, then* oCMZ *satisfies attribute-based unforgeability in the random oracle model and the algebraic group model.*

Proof. The proof is by reduction to the unforgeability of $\mathsf{MAC_{GGM}}$, with one hybrid change strengthening the game $\mathrm{AUF}_{\mathsf{oCMZ,A}}(\lambda)$.

We consider a p.p.t. adversary A for the game $\mathrm{AUF}_{\mathsf{oCMZ,A}}(\lambda)$ that takes as input the pair (crs, pk) and returns a forgery σ^* for the message $\boldsymbol{m}^*$. We add one extra winning condition: in addition $\boldsymbol{m}^* \notin Q$ and $\mathsf{oCMZ.Ver}(crs, \boldsymbol{X}, \boldsymbol{m}^*, \sigma^*) = \mathbf{true}$, we demand that also secret-key verification is satisfied, that is:

$$\mathsf{oCMZ.sVer}(crs, \boldsymbol{x}, \boldsymbol{m}^*, \sigma^*) = \mathbf{true}.$$

This game can be distinguished with at most negligible probability if oTZ for the relation $\mathsf{R}_{\mathrm{CMZ}}$ is adaptive sound. Consider a p.p.t. adversary B for the game $\mathrm{SND}_{\mathsf{oTZ,B}}(\lambda)$, that, given as input (Γ, Λ) samples $H \leftarrow\!\!\$\ \mathbb{G}$ and sets $crs = (\Gamma, H, \Lambda)$. Then, sets $(\boldsymbol{x}, \boldsymbol{X}) \leftarrow \mathsf{oCMZ.KeyGen}(crs)$ and internally runs $\mathsf{A}(crs, \boldsymbol{X})$, replying to the oracle queries just as illustrated in Figure 12. Upon receiving the output of A, that is: $\boldsymbol{m}^*, (Y^*, Z^*), (a^*, b^*, e^*, \boldsymbol{r}^*)$, B outputs the witness $\boldsymbol{x}$, the statement $(\boldsymbol{X}, \boldsymbol{Y}^* = (\boldsymbol{m}^* \cdot Y), Z^*)$ and the proof $(a^*, b^*, e^*, \boldsymbol{r}^*)$. The two games have a different outcome only if the verification algorithm suceeds and yet the secret-key verification algorithm doesn't. In other words, if:

$$(Y^*, Z^*) \neq (Y^*, (x_0 + \textstyle\sum_i x_i m_i) Y^*)$$

which in turn implies that $(\boldsymbol{X}, \boldsymbol{Y}, Z) \notin \mathsf{R}_{\mathrm{CMZ}}$ and $\mathsf{oCMZ.Ver}$ succeds. Therefore, B has returned a valid forgery.

From the above game, we can build a reduction B that will act as an attacker in the $\mathsf{uf\text{-}cmva}$ game for $\mathsf{MAC_{GGM}}$, and use the attribute-based forger A as a subroutine (A is the attacker in Figure 12). As part of the oCMZ setup, B constructs the committer key ck by sampling $h \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and computing $H := hG$, and can therefore equivocate the commitment C. For the i-th signing query $\mathrm{ISSUER}_1(\boldsymbol{m}_i)$ for a message vector $\boldsymbol{m}_i$, B queries the MAC oracle of the $\mathsf{uf\text{-}cmva}$ game with $\boldsymbol{m}$, obtaining a MAC (Y', Z') on $\boldsymbol{m}_i$. Then, B samples $\hat{a}_i \leftarrow\!\!\$\ \mathbb{Z}_p^*$, $\hat{b}_i \leftarrow\!\!\$\ \mathbb{Z}_p$, and $\boldsymbol{r}_i \leftarrow\!\!\$\ \mathbb{Z}_p^n$ and computes $\boldsymbol{T}_i' := [\boldsymbol{Y}_i'; M]\boldsymbol{r}_i' - \hat{a}_i \boldsymbol{X}$, and $C' := \hat{a}_i H + \hat{b}_i G$. Upon receiving queries of the form $\mathrm{ISSUER}_2(i, e_i')$, B equivocates the commitment setting $a' := \hat{a}_i / e_i'$ and $b' := \hat{b}_i - (\hat{a}_i - a_i') \cdot h$ and returns $((a', b'), \boldsymbol{T}', e', r')$. For any new query to the random oracle of the form $\mathrm{H}((0, Y, \boldsymbol{m} \cdot Y), Z, \boldsymbol{T}, C)$ if the oracle was previously queried on the same input, then B returns the previously-stored value; otherwise it samples a fresh challenge $e \leftarrow\!\!\$\ \mathbb{Z}_p$, internally stores it in a hash table with key $(\boldsymbol{Y}, Z, \boldsymbol{T}, C)$, and returns e. Any query to the verification oracle VERIFY is forwarded to the verification oracle of the $\mathsf{uf\text{-}cmva}$ game.

```
Game $\mathrm{AUF}_{\mathsf{MAC},\mathsf{A}}(\lambda)$
    $crs \leftarrow \mathsf{MAC.Setup}(1^\lambda)$
    $(sk, pk) \leftarrow \mathsf{MAC.KeyGen}(crs)$
    $\ell := 0;\ Q := \emptyset;\ Opn := \emptyset$
    $(\sigma^*, \boldsymbol{m}^*) \leftarrow \mathsf{A}^{\mathrm{H},\textsc{Issuer}_1,\textsc{Issuer}_2,\textsc{Verify}}(crs, pk)$
    return $(\mathsf{MAC.Ver}(crs, pk, \boldsymbol{m}^*, \sigma^*) = \mathbf{true}$
        and $\boldsymbol{m}^* \notin Q)$

Oracle $\textsc{Verify}(\mu, \boldsymbol{m})$
    return $\mathsf{MAC.Ver}(crs, sk, \mu, \boldsymbol{m})$

Oracle $\textsc{Issuer}_1(\boldsymbol{m})$
    $i := i + 1;\ Opn := Opn \cup \{i\}$
    $Q := Q \cup \{\boldsymbol{m}\}$
    $(\pi_1, st_i) \leftarrow \mathsf{MAC.Sign}_1(crs, sk, pk, \boldsymbol{m})$
    return $\pi_1$

Oracle $\textsc{Issuer}_2(i, e_i')$
    if $i \notin Opn$ return $\perp$
    $\ell := \ell + 1;\ Opn := Opn \setminus \{i\}$
    $\pi_2 \leftarrow \mathsf{MAC.Sign}_2(st_i, e_i')$
    return $\pi_2$
```

Fig. 12. The attribute unforgeability (AUF) game for blind signatures from algebraic MACs. H is the random oracle with range over $\mathbb{Z}_p$.

At the end of its execution, the adversary A returns a signature $(Y^*, Z^*), (a^*, b^*, e^*, \boldsymbol{r}^*)$ and a message $\boldsymbol{m}^*$ such that $\mathsf{MAC.sVer}(crs, \boldsymbol{x}, \boldsymbol{m}^*, (Y^*, Z^*)) = \mathbf{true}$ and which therefore constitutes a forgery for $\mathsf{MAC}_{\mathrm{GGM}}$.

The following lemma states that the matrix of Equation (10) is such that KMDH is as hard as DL: in fact, finding a non-zero $\boldsymbol{r}$ such that $M\boldsymbol{r} = \mathbf{0}G$ implies solving the DL of W to base G (cf. second row): Since $M\boldsymbol{r} = 0$, then $r_0 \neq 0$ or $r_1 \neq 0$ (otherwise one of the rows $1..n$ is non-zero) and the reduction to DL is immediate.

Lemma 11. *If DL is hard for* GrGen, *then for any p.p.t. adversary* A,

$$\mathsf{Adv}^{\mathrm{kmdh}}_{\mathsf{GrGen},\mathsf{R}_{\mathrm{CMZ}},\mathsf{A}}(\lambda) = \mathsf{Adv}^{\mathrm{dl}}_{\mathsf{GrGen},\mathsf{A}}(\lambda) \ .$$

Corollary 3. *If* $\mathsf{MAC}_{\mathsf{GGM}}$ *is* uf-cmva *secure [15, Def. 1] and DL is hard for* GrGen, *then* oCMZ *is unforgeable for* $n \in \mathsf{poly}(\lambda)$ *attributes.*

Remark 3 (Somewhat selective disclosure). So far we have considered public verification of oCMZ when the verifier is given all of the attribute values. This is the behavior of blind signatures, but as a single-use unlinkable credential with public verifiability oCMZ is less capable than alternatives like AC Light and U-Prove, which can *selectively disclose* subsets of the attributes, and keep undisclosed attributes unconditionally hidden. Referring to $\mathsf{oCMZ.Ver}$ (Figure 11), we see that of the inputs the verifier gets, only Z depends on the attributes $\boldsymbol{m}$. We first note that Z hides $\boldsymbol{m}$ when one of the attributes is uniformly random and kept hidden (this is the case even when the verifier knows the secret key $\boldsymbol{x}$). Now we argue that verification remains correct Now consider $\mathsf{oCMZ.Ver}'$, where we replace $\boldsymbol{m}$ by $\boldsymbol{Y} = (0, Y, m_1 Y, m_2 Y, \ldots, m_n Y)$. Since the first step of $\mathsf{oCMZ.Ver}$ is to compute $\boldsymbol{Y}$, $\mathsf{oCMZ.Ver}'$ remains correct if $\boldsymbol{Y}$ is well-formed. Thus if we additionally require a knowledge-sound proof π_Y that $\boldsymbol{Y}$ is well-formed, calls to $\mathsf{oCMZ.Ver}'$ can be reduced directly to $\mathsf{oCMZ.Ver}$ by extracting $\boldsymbol{m}$ from π_Y and calling $\mathsf{oCMZ.Ver}$. Unfortunately, $m_i Y$ is not perfectly hiding, so using $\mathsf{oCMZ.Ver}'$ still reveals some attribute information, however, this is sufficient for use-cases where attributes have high entropy. When deploying oCMZ it will usually be essential to have at least one attribute be random, kept private during issuance (Section 6.3) and always kept hidden; otherwise $\mathsf{oCMZ.Ver}$ obtains the MAC value (Y, Z) and the attributes $\boldsymbol{m}$, they can then start presenting the credential unlinkably as a MAC. Designing a richer credential system on top of oCMZ is interesting future work.

Signature size. The signature size is $2|\mathbb{G}| + (3 + n)|\mathbb{Z}_p|$ bits, which is linear in the number of attributes (as is usually the case for attribute-based credentials). To put this in perspective, we can compare to other credentials constructed in prime order groups, in particular AC Light and U-Prove. We caveat that the comparison is not direct since neither supports the keyed-verification feature of our scheme, and the public verification in our scheme does not (directly) support the rich presentation proofs of the other two. Finally, the security properties of all three schemes vary.

We compare the size of presenting a credential by counting the number of elements, assuming for simplicity that $|\mathbb{G}| \approx |\mathbb{Z}_p|$ (as in elliptic curve groups). In AC Light, the signature size is eight elements, to show that a ninth is a fresh commitment to the n attributes. Therefore, the cost of credential presentation when all attributes are revealed is $9+n$ elements. For U-Prove the cost is $6+n$ elements. Both are slightly more than the $5+n$ elements required by our scheme.

Remark 4. An interesting question is to what extent our approach translates to other algebraic MACs used to construct KVAC schemes such as in [4,11,16]. For example, the same idea easily translates to $\mathsf{MAC}_{\mathsf{DDH}}$ [15, Section 3.2, 4.3], and we expect the security analysis to be similar. When comparing $\mathsf{MAC}_{\mathrm{DDH}}$ and $\mathsf{MAC}_{\mathrm{GGM}}$, the main difference is that the witness becomes a matrix instead of a simple vector. The secret-key is defined as

$$(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) \coloneqq ((x_{-1}, x_0, \ldots, x_n), (u_{-1}, u_0, \ldots, u_n), (v_{-1}, v_0)) \leftarrow\!\!\$\ \mathbb{Z}_p^{n+2} \times \mathbb{Z}_p^{n+2} \times \mathbb{Z}_p^2$$

while the public key is

$$\begin{aligned} \boldsymbol{X} &= (x_0 G + x_{-1} W, x_1 W, x_2 W, \ldots, x_n W), \\ \boldsymbol{U} &= (u_0 G + u_{-1} W, u_1 W, u_2 W, \ldots, u_n W), \\ \boldsymbol{V} &= (v_0 G + v_{-1} W), \end{aligned}$$

A MAC is computed as:

$$(\mu_0, \mu_1, \mu_2, \mu_3) \coloneqq (Y, (x_0 + \sum_i m_i x_i) Y, (u_0 + \sum_i m_i u_i) Y, v_0 Y)$$

which, written in the language of generic algebraic relations when instantiating our generic protocol is:

$$\begin{bmatrix} \boldsymbol{Y} \\ M \end{bmatrix} \begin{bmatrix} \boldsymbol{x} & \boldsymbol{u} & \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} 0 & Y & m_1 Y & m_2 Y & \cdots & m_n Y \\ W & G & 0 & 0 & \cdots & 0 \\ 0 & 0 & W & 0 & \cdots & 0 \\ 0 & 0 & 0 & W & \cdots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & W \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_{-1} & u_{-1} & v_{-1} \\ x_0 & u_0 & v_0 \\ x_1 & u_1 & 0 \\ x_2 & u_2 & 0 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 0 \end{bmatrix} = \begin{bmatrix} \mu_1 & \mu_2 & \mu_3 \\ \boldsymbol{X} & \boldsymbol{Y} & \boldsymbol{Z} \end{bmatrix}$$

6.3 Supporting private attributes

In [15], an alternative issuance protocol is described that allows the user to keep some attributes private during issuance. This feature is commonly referred to as "blind issuance" in the literature on anonymous credentials. It is possible to extend oCMZ issuance support private attributes as well. Let $\mathit{Hid} \subset [1, n]$ denote the set of indices of the attributes to be kept private, and consider a linearly-homomorphic encryption scheme over Γ. For the rest of this section, we set the scheme to be Elgamal, and denote it as $\mathcal{E}$. The user generates an Elgamal keypair $(s \in \mathbb{Z}_p, S = sG \in \mathbb{G}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(\mathit{crs})$ and computes an Elgamal encryption of each $i \in \mathit{Hid}$

$$\mathit{ctx}_{m_i} \coloneqq (E_{i,1}, E_{i,2}) \coloneqq (r_i G, m_i G + r_i S) \leftarrow \mathcal{E}.\mathsf{Enc}(S, m_i)$$

with $r_i \leftarrow\!\!\$\ \mathbb{Z}_p$. For simplicity, we denote the vector encryption of $\boldsymbol{m}_{\text{hid}} = \{m_i\}_{i \in \mathit{Hid}}$ as $\mathit{ctx}_{\boldsymbol{m}_{\text{hid}}}$. Instead of sending the plain vector of messages $\boldsymbol{m}$ in the algorithm $\mathsf{oCMZ.Usr}_0$, now the user sends the ciphertexts $\mathit{ctx}_{\boldsymbol{m}_{\text{hid}}} = \{(E_{i,1}, E_{i,2})\}_i$ to the issuer along with a zero-knowledge proof of knowledge of (r_i, m_i).[9] The issuer, instead of computing $\boldsymbol{Y}', Z'$ according to the protocol description above, samples $y' \leftarrow\!\!\$\ \mathbb{Z}_p$ and

[9] In this step, the user generally also proves that m_i satisfies some constraint, e.g. is of 8 bits.

```
$\mathsf{oCMZ.Usr}(crs, \boldsymbol{X}, (Hid, \boldsymbol{m}_{\text{pub}}, \boldsymbol{m}_{\text{hid}}))$                    $\mathsf{oCMZ.Iss}(crs, \boldsymbol{x})$

$(s, S) \leftarrow \mathcal{E}.\mathsf{KeyGen}(crs)$
$ctx_{\boldsymbol{m}_{\text{hid}}}, \boldsymbol{\pi}_{\text{hid}} := \mathcal{E}.\mathsf{Enc}(s, \boldsymbol{m}_{\text{hid}})$
        $\xrightarrow{S, ctx_{\boldsymbol{m}_{\text{hid}}}, \boldsymbol{\pi}_{\text{hid}}, \boldsymbol{m}_{\text{pub}}}$
                                        $y' \leftarrow\!\!\$\ \mathbb{Z}_p;\quad Y' := y'G$
                                        // If $Hid = \emptyset$, compute and send
                                        // $Z' := (x_0 + \langle \boldsymbol{x}_{1..n}, \boldsymbol{m}_{\text{pub}} \rangle) \cdot Y'$
                                        // Otherwise:
                                        Hom. compute $ctx_{Z'}$ as per Equation (11)
        $\xleftarrow{Y', ctx_{Z'}}$
$\upsilon \leftarrow\!\!\$\ \mathbb{Z}_p$
$Y := \upsilon^{-1} Y'$
// If $Hid = \emptyset$, skip this step
$Z' := \mathcal{E}.\mathsf{Dec}(s, ctx_{Z'})$
$Z := \upsilon^{-1} Z'$
                                        $\boldsymbol{t}' \leftarrow\!\!\$\ \mathbb{Z}_p^{n+2}$
                                        // If $Hid = \emptyset$, compute and send
                                        // $\boldsymbol{T}' := [\boldsymbol{Y}'; M] \cdot \boldsymbol{t}'$
                                        // Otherwise:
                                        Hom. compute $ctx_{T'_0}$ as per Equation (12)
                                        Compute $\boldsymbol{T}'_{1..n+1}$ as per Equation (13)
                                        $(a', b') \leftarrow\!\!\$\ \mathbb{Z}_p^2$
                                        $C' := a'H + b'G$
        $\xleftarrow{ctx_{T'_0}, \boldsymbol{T}'_{1..n}, C'}$
// If $Hid = \emptyset$, skip this step
$\boldsymbol{T}' := [\mathcal{E}.\mathsf{Dec}(s, ctx_{T'_0}); \boldsymbol{T}'_{1..n+1}]$
$\varepsilon \leftarrow\!\!\$\ \mathbb{Z}_p;\quad \boldsymbol{\rho} \leftarrow\!\!\$\ \mathbb{Z}_p^{n+2}$
$\boldsymbol{T} := \varepsilon^{-1} \begin{bmatrix} \upsilon^{-1} & 0 & \cdots \\ 0 & 1 & \cdots \\ & \cdots & \mathbf{1} \end{bmatrix} \cdot \left( \boldsymbol{T}' - \begin{bmatrix} \boldsymbol{Y}' \\ M \end{bmatrix} \boldsymbol{\rho} \right)$
$(\alpha, \beta) \leftarrow\!\!\$\ \mathbb{Z}_p^2$
$C := \alpha^{-1} C' - \beta G$
$e := \mathrm{H}(Y, Z, \boldsymbol{T}, C)$
$e' := \varepsilon \alpha^{-1} e$
        $\xrightarrow{e'}$
                                        $\boldsymbol{r}' := \boldsymbol{t}' + e'a' \cdot \boldsymbol{x}$
        $\xleftarrow{\boldsymbol{r}', a', b'}$
**check** $Y' \neq 0G$
**check** $[\boldsymbol{Y}'; M] \cdot \boldsymbol{r}' = e'a' \cdot [Z'; X_0; X_1; \dots; X_n] + \boldsymbol{T}'$
**check** $C' = a'H + b'G$ **and** $a' \neq 0$
$\boldsymbol{r} := \varepsilon^{-1}(\boldsymbol{r}' - \boldsymbol{\rho})$
$a := \alpha^{-1} a'$
$b := \alpha^{-1} b' - \beta$
**return** $(Y, Z), (a, b, e, \boldsymbol{r})$
```

Fig. 13. Issuance protocol for algebraic MAC-based blind signatures. $\mathcal{E}$ denotes the Elgamal encryption scheme.

computes $Y' := y'G$ and computes an encryption of Z' using the linearity of the encryption scheme. Sample $a \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute

$$ctx_{Z'} := (E'_1, E'_2) := (aG + \textstyle\sum_{i \in Hid} x_i y' E_{i,1},\ aS + x_0 y' G + \sum_{i \in Hid} x_i y' E_{i,2} + \sum_{i \in [1,n] \setminus Hid} m_i x_i y' G). \tag{11}$$

The server also proves that the ciphertext is well-formed: the resulting ciphertext is $(E'_1, E'_2 = E'_1 + x_0 Y' + \langle \boldsymbol{x}_{1..n}, \boldsymbol{m} \rangle Y')$, which is a valid encryption of Z'. The user decrypts Z' and stores it internally.

Despite it is not possible to compute the matrix Y' directly, it is still possible to compute the matrix-vector product using (again) linearity of $\mathcal{E}$. Given a list of index attributes Hid, an encryption of the element T'_1 is computed as follows:

$$ctx_{T'_1} := \mathcal{E}.\mathsf{Enc}(S, T'_1) := (aG + \textstyle\sum_{i \in Hid} t_i y' E_{i,1},\ aS + t_0 y' G + \sum_{i \in Hid} t_i y' E_{i,2} + \sum_{i \in [1,n] \setminus Hid} m_i t_i y' G). \tag{12}$$

The remaining elements $\boldsymbol{T}'_{2..n+2}$ are computed using the rows $[2, n+1]$ of Equation (10)

$$\boldsymbol{T}'_{1..n+2} \coloneqq \begin{bmatrix} W & G & 0 & 0 & \cdots & 0 \\ 0 & 0 & W & 0 & \cdots & 0 \\ 0 & 0 & 0 & W & \cdots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & W \end{bmatrix} \boldsymbol{t}' \tag{13}$$

Since Elgamal is perfectly correct, correctness follows immediately from the correctness of the oCMZ protocol.

Unlinkability follows straightforwardly from the argument of the previous section, since the ciphertexts $\mathcal{E}.\mathsf{Enc}(S, \boldsymbol{m}_{\text{priv}})$ are indistinguishable from uniformly random samples over $\mathbb{G}^2$ if DDH is hard for GrGen, by IND-CPA of Elgamal encryption. The tuple $(\mu', \mu) = (Y', Z', Y, Z) = (\upsilon Y, \upsilon Z, Y, Z)$ is indistinguishable from uniformly random if DDH is hard for GrGen. (Note: differently from the previous proof, here the messages may be different.)

A similar reasoning to the analysis of the previous section also applies to one-more unforgeability: since all private attributes are followed by a valid zero-knowledge proof of knowledge, it is always possible to extract the private messages from the proof thus reducing to the (simpler) experiment with public attributes only. Similarly, attribute unforgeability still holds when some attributes are hidden. Consider an adversary A that, after interacting ℓ times with the issuer, outputs a single signature $(Y^*, Z^*), (a^*, b^*, e^*, \boldsymbol{r}^*)$ and a message vector $\boldsymbol{m}$ that has not been signed before. Considering [3], this game is stronger than Definition 7, where only the multiset of public attributes are taken into account. However, we can extract the private messages from the proof and any forgery for a private attribute constitutes a break for soundness.

7 A secure variant of U-Prove

U-Prove is a cryptographic protocols initially designed by Stefan Brands that allows users to minimally disclose information about what attributes are encoded in a token. Each token is unlinkable, preventing tracking of users. We focus on the issuance of a U-Prove token, which is illustrated in [36, Fig. 7]. We provide a mitigation for the attack from Benhamouda et al. [6] for the case where U-Prove tokens are generated using identical common inputs and the computation is shared among parallel protocol executions. The resulting protocol is almost identical to the previous one, except for the additional elements in the proof transcript arising from the Tessaro-Zhu transform. The tokens themselves are left unchanged.

7.1 Syntax and Definition

A abstraction of U-Prove issuance protocol consists of a tuple of p.p.t. algorithms $\mathsf{UProve} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Iss}, \mathsf{Usr}, \mathsf{Ver})$, with the following functionalities:

- $\mathit{crs} \leftarrow \mathsf{UProve.Setup}(1^\lambda)$, the setup algorithm, generates the public parameterscrs
- $(\mathit{sk}, \mathit{pk}) \leftarrow \mathsf{UProve.KeyGen}(\mathit{crs})$, the key generation algorithm, generates a *secret key* sk and a *public verification key* pk
- The interactive algorithms $\mathsf{UProve.Iss}$ (the *issuing* algorithm) and $\mathsf{UProve.Usr}$ (the *user* algorithm) take as input $(\mathit{sk}, \mathit{pk})$, and $(\mathit{pk}, \boldsymbol{m})$, respectively, along with crs. The interaction and the outputs of the issuer and user are denoted as:

  $$(\mathsf{cred}, \bot) \leftarrow \langle \mathsf{UProve.Usr}(\mathit{crs}, \mathit{pk}, \boldsymbol{m}), \mathsf{UProve.Iss}(\mathit{crs}, \mathit{sk}, \mathit{pk}, \boldsymbol{m}) \rangle$$

  where $\boldsymbol{m} \in (\mathcal{M})^n$ denotes a list of n attributes each from a set $\mathcal{M}$ and cred denotes the resulting token.
- **true**/**false** $\leftarrow \mathsf{UProve.Ver}(\mathit{crs}, \mathit{pk}, \boldsymbol{m}, \mathsf{cred})$ outputs a bit to indicate whether cred is a valid token for attributes $\boldsymbol{m}$.

$\mathsf{UProve.Usr}(crs, pk, \boldsymbol{m})$		$\mathsf{UProve.Iss}(crs, sk, pk, \boldsymbol{m})$
	$\xrightarrow{\boldsymbol{m}}$	
	$\xleftarrow{Z', \boldsymbol{T}', C'}$	$st, (Z', \boldsymbol{T}', C') \leftarrow \mathsf{oTZ.Iss}_1(crs, sk, \boldsymbol{m})$
$st, e' \leftarrow \mathsf{oTZ.Usr}_1(crs, pk, \boldsymbol{m}, (Z', \boldsymbol{T}', C'); \upsilon)$	$\xrightarrow{e'}$	
	$\xleftarrow{r', a', b'}$	$(r', a', b') \leftarrow \mathsf{oTZ.Iss}_2(st, e')$
$((Y, Z), (a, b, e, r)) \leftarrow \mathsf{oTZ.Usr}_2(st, (r', a', b'))$		
return $(\upsilon, Y, Z, (a, b, e, r))$		

Fig. 14. A secure variant of U-Prove issuance protocol.

We consider only one-more unforgeability for UProve here, since it is the only security guarantee affected by the ROS attack. Informally, one-more unforgeability is defined by the following security game. The security game initially runs $\mathsf{UProve.Setup}(1^\lambda)$ to generate *crs* and $(sk, pk) \leftarrow \mathsf{UProve.KeyGen}(crs)$. The p.p.t. adversary, given *crs* and *pk*, can then interact concurrently over ℓ sessions with $\mathsf{UProve.Iss}(crs, sk)$. It wins if it outputs $\ell+1$ distinct pairs $\{(\boldsymbol{m}_j, \mathsf{cred}_j)\}_{j \in [\ell+1]}$ such that $\mathsf{UProve.Ver}(crs, pk, m_j, \mathsf{cred}_j) = \mathbf{true}$ for all $j \in [\ell + 1]$.

7.2 Our Construction

Our construction UProve is obtained from $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ for the following DLEQ relation:

$$\mathsf{R}_{\mathrm{up}} := \left\{ (x, X, Y, Z) : x \in \mathbb{Z}_p, Y \in \mathbb{G}, \quad x \begin{bmatrix} Y \\ G \end{bmatrix} = \begin{bmatrix} Z \\ X \end{bmatrix} \right\},$$

The relation specific setup $\mathsf{R}_{\mathrm{up}}.\mathsf{Setup}(\Gamma)$ returns generators $\boldsymbol{G}$ with $\boldsymbol{G} := (G_0, G_1, \ldots, G_n) \in \mathbb{G}^l$ whose discrete logarithm is not known. The setup algorithm for the overall protocol outputs $(\Gamma, H, \boldsymbol{G}) \leftarrow \mathsf{UProve.Setup}(\Gamma)$. We regard $\boldsymbol{m} \in \mathbb{Z}_p^n$ as *info*, and the $\mathsf{R}_{\mathrm{up}}.\mathsf{SampleArg}$ is deterministic algorithm that takes $(crs, \boldsymbol{m} \in \mathbb{G}^n)$ as input and outputs $Y' = \sum m_i G_i + G_0$. Then, the protocol UProve is constructed as follows.

- The setup algorithm UProve.Setup is the same as the setup algorithm of $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ for R_{up}.
- The key generation algorithm UProve.KeyGen, given *crs*, samples $sk \leftarrow\!\!\$\ \mathbb{Z}_p$ and sets $pk \leftarrow sk \cdot G$.
- The issuance algorithms UProve.Iss and UProve.Usr are the same as those of $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}]$ except the user also outputs the random coins υ generated by the user protocol together with the tuple $(Y, Z, \pi = (a, b, e, r))$. We also present the algorithms in Figure 14.
- The verification algorithm, given *pk*, $\boldsymbol{m}$ and $\mathsf{cred} = (\upsilon, Y, Z, \pi)$, first checks $Y = \upsilon(\sum m_i G_i + G_0)$ and then outputs $\mathsf{oTZ}[\mathsf{GrGen}, \mathrm{restr}].\mathsf{Ver}(pk, Y, Z, \pi)$.

Highlighted in orange the changes with the cryptographic specification.[10]

The pair (υ, Y) is the *token private and public key*, which should be unique to each token, and required to present the token. The stated purpose of the token private key is to prevent replay of the token. One-more unforgeability of UProve follows immediately from one-more unforgeability of the underlying oTZ protocol.

References

1. Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, Heidelberg, May 2001.

[10] The following change of variables have been applied to maintain consistency with additive groups and the notation used in this work: $y_0 = x$; $\upsilon = Y$; $\sigma_z, \sigma_z' = Z', Z$; $\sigma_c, \sigma_c' = e', e$; $\sigma_r, \sigma_r' = r', r$; $\beta_1 = \varepsilon$; $\beta_2 = \rho$; $\alpha = \upsilon$; $[\sigma_a; \sigma_b], [\sigma_a'; \sigma_b'] = \boldsymbol{T}', \boldsymbol{T}$. The challenge e (σ_c in the spec) is now masked multiplicatively to be compatible with the TZ technique.

2. Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286. Springer, Heidelberg, August 2000.
3. 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.
4. Amira Barki, Solenn Brunet, Nicolas Desmoulins, Sébastien Gambs, Saïd Gharout, and Jacques Traoré. Private eCash in practice (short paper). In Jens Grossklags and Bart Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 99–109. Springer, Heidelberg, February 2016.
5. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.
6. 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.
7. 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.
8. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.
9. Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2020. Available online https://toc.cryptobook.us/book.pdf.
10. Stefan Brands. *Rethinking public key infrastructures and digital certificates: building in privacy*. Mit Press, 2000.
11. Jan Camenisch, Manu Drijvers, Petr Dzurenda, and Jan Hajny. Fast keyed-verification anonymous credentials on standard smart cards. In *ICT Systems Security and Privacy Protection*, pages 286–298. Springer, 2019.
12. Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, 260, 1997.
13. Sofia Celi, Alex Davidson, Armando Faz-Hernandez, Steven Valdez, and Christopher A. Wood. Privacy Pass Issuance Protocol. Internet-Draft draft-ietf-privacypass-protocol-10, Internet Engineering Task Force, March 2023. Work in Progress.
14. Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Heidelberg, August 2022.
15. Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 1205–1216. ACM Press, November 2014.
16. Melissa Chase, Trevor Perrin, and Greg Zaverucha. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1445–1459. ACM Press, November 2020.
17. 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.
18. Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI Amsterdam, The Netherlands, 1997.
19. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.
20. Alfredo De Santis and Moti Yung. Crptograpic applications of the non-interactive metaproof and many-prover systems. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 366–377. Springer, Heidelberg, August 1991.
21. Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. Internet-Draft draft-irtf-cfrg-rsa-blind-signatures-12, Internet Engineering Task Force, April 2023. Work in Progress.
22. Florian Dold. *The GNU Taler system : practical and provably secure electronic payments*. Theses, Université de Rennes, February 2019.
23. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.
24. 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.
25. Georg Fuchsbauer and Mathias Wolf. (concurrently secure) blind schnorr from schnorr. Cryptology ePrint Archive, Paper 2022/1676, 2022. https://eprint.iacr.org/2022/1676.

26. Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 228–245. Springer, Heidelberg, August 1993.
27. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.
28. Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Rai-choo! Evolving blind signatures to the next level. Cryptology ePrint Archive, Report 2022/1350, 2022. https://eprint.iacr.org/2022/1350.
29. Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Heidelberg, May 2019.
30. 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.
31. Julia Kastner, Julian Loss, Michael Rosenberg, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. Cryptology ePrint Archive, Report 2020/1071, 2020. https://eprint.iacr.org/2020/1071.
32. Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 468–492. Springer, Heidelberg, December 2021.
33. Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.
34. Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016.
35. 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.
36. Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1. 1, revision 5. *Technical Report, Microsoft Corporation*, 2011.
37. David Pointcheval. Strengthened security for blind signatures. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 391–405. Springer, Heidelberg, May / June 1998.
38. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.
39. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.
40. Claus-Peter Schnorr. Efficient identification and signatures for smart cards (abstract) (rump session). In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT'89*, volume 434 of *LNCS*, pages 688–689. Springer, Heidelberg, April 1990.
41. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.
42. 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.
43. Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 782–811. Springer, Heidelberg, May / June 2022.
44. Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious PRF, with applications. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 674–705. Springer, Heidelberg, May / June 2022.
45. David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002.