Improved Constructions of Anonymous Credentials From Structure-Preserving Signatures on Equivalence Classes *

Aisling Connolly[1], Pascal Lafourcade[2], and Octavio Perez Kempner[3,4]

[1] Worldline Global
aislingmconnolly@gmail.com
[2] LIMOS, University Clermont Auvergne, France
pascal.lafourcade@uca.fr
[3] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[4] be-ys Research, France
octavio.perez.kempner@ens.fr

Abstract. Anonymous attribute-based credentials (ABCs) are a powerful tool allowing users to authenticate while maintaining privacy. When instantiated from structure-preserving signatures on equivalence classes (SPS-EQ) we obtain a controlled form of malleability, and hence increased functionality and privacy for the user.
Existing constructions consider equivalence classes on the message space, allowing the joint randomization of credentials and the corresponding signatures on them. In this work, we additionally consider equivalence classes on the signing-key space. In this regard, we obtain a *signer-hiding* notion, where the issuing organization is not revealed when a user shows a credential. To achieve this, we instantiate the ABC framework of Fuchsbauer, Hanser, and Slamanig (FHS19, Journal of Cryptology '19) with a recent SPS-EQ scheme (ASIACRYPT '19) modified to support a fully adaptive NIZK from the framework of Couteau and Hartmann (CRYPTO '20). We also show how to obtain mercurial signatures (CT-RSA '19), extending the application of our construction to anonymous delegatable credentials.
To further increase functionality and efficiency, we augment the set-commitment scheme of FHS19 to support openings on attribute sets disjoint from those possessed by the user, while integrating a proof of exponentiation to allow for a more efficient verifier. Instantiating in the CRS model, we obtain an efficient credential system, anonymous under malicious organization keys, with increased expressiveness and privacy, proven secure in the standard model.

Keywords: Anonymous credentials · Mercurial signatures · SPS-EQ

Erratum: After publication, it was observed that our signature construction provides signer-hiding under *honestly* generted keys in the honest parameter model with respect to the key space (and not under *maliciously* generated keys as originally claimed). We thank Colin Putman for this observation.

1 Introduction

Considering access to online services, designing protocols to manage the information users can be requested to present is of utmost importance to protect the user. A first step in the literature developed the concept of *attribute-based credentials* (ABC), to model how users could show a credential, containing a set of attributes, to access different services.

Subsequently, the development of *anonymous* attribute-based credentials made it possible protect the holders identity when showing a credential. Users could present a credential disclosing no information other than that revealed by the attributes they choose to show (*anonymity*), while also ensuring that the provided information is authentic (*unforgeability*). Proposed alternatives consider a third property *unlinkability* which ensures that multiple showings of the same credential cannot be linked. Credential systems that support an arbitrary number of unlinkable showings are said to be *multi-show*. In contrast, those that only allow a single use of an issued credential in an unlinkable fashion are called *one-show*.

Initial progress was made with respect to one-show constructions. Here, *blind signatures* are issued on commitments to attributes so that users can later show the signature and disclose some of

* This is the full version of [CLPK22]. In case of citing our work, please cite the proceedings version.

the attributes, while proving knowledge of those left unrevealed. Examples include [Bra00, BL13], and [FHKS16].

In the multi-show setting, pioneering constructions (based on Camenisch and Lysyanskaya's (CL) signatures [CL02, CL04]) such as the one underlying the Idemix credential system [Zur13] rely on randomizing the signature to then prove in zero-knowledge the correspondence between the set of attributes (disclosed and undisclosed), and the signature. A major drawback from such an approach is that the zero-knowledge proof used during showings is of variable-length and may require multiple sub-proofs On the other hand, more recent constructions (*e.g.,* [CL11, CDHK15, San20, HP20, TG20, DHS15a, FHS19]) apply other techniques based on different lines of work to adapt the signature and the message without using Zero-Knowledge Proofs of Knowledge (ZKPoK), providing constant-size showings.

The concept of ABC has been recently extended to consider multi-authority credentials (*e.g.,* [HP20, SABB+19]), where users obtain a single credential for a set of attributes not necessarily issued by a single authority. In this work we consider the *classical* setting (single authority issuance).

1.1 Limitations of state-of-the-art ABCs

Constructions in the classical setting differentiate from each other by the expressiveness they provide, their efficiency, on whether or not they provide non-interactive features, on their security model, and on how and if they manage revocation features. Achieving all these properties simultaneously has been challenging and tends to rely on complex or non-standard assumptions.

When considering state-of-the-art credential systems, there are five lines of work with respect to the underlying signature scheme that is used to build them; (1) CL signatures [CL04]: Idemix [Zur13] and [TG20]. (2) Aggregatable signatures: [CL11] and [HP20]. (3) Sanitizable signatures: [CL13]. (4) Redactable signatures: [CDHK15] and [San20]. (5) Structure-Preserving Signatures on Equivalence Classes (SPS-EQ): [FHS19].

Proof settings. All previous work with the exception of [TG20] rely on security proofs in the Generic Group Model (GGM) [Sho97]. Our first motivation is to provide an alternative to [TG20], building on [FHS19] without relying on the GGM.

Signer-hiding properties. Showing protocols of previous constructions (including [TG20]), verify signatures with a key that belongs to the authority that issued the credential. This restricts the use of ABC in scenarios where one would like to verify a valid credential without linking it to a particular authority.

Concrete efficiency. Most alternatives provide similar efficiency at the asymptotic level. Yet, an up-to-date fine-grained analysis on their concrete efficiency lacks in the literature.

1.2 Summary of contributions

We follow the ABC and SPS-EQ line of work from Fuchsbauer, Hanser and Slamanig [FHS19], improving over prior work in the following ways:

1. We extend the set-commitment scheme from [FHS19] to build a more expressive credential system allowing the generation of witnesses for disjoint sets ([FHS19] allows only selective disclosure of attributes).
2. We instantiate the ABC from [FHS19], with a new SPS-EQ scheme based on the one from [KSD19] also using a CRS, a tight reduction, and under weaker assumptions. Thus, we move away from a security proof in the GGM when compared to the work from [FHS19], and obtain a more efficient ABC than the one resulting from instantiating [FHS19] with [KSD19] (see Table 1).
3. We incorporate a proof of exponentiation to outsource part of computational cost from the verifier to the prover, which can be useful in some settings.
4. We adapt the signature scheme to build an SPS-EQ where one not only can randomize the message together with the signature, but also the corresponding public key used to verify the signature using a proof of well-formedness. Thus, users can hide the identity of the *signer* during showings.

By doing so, the verifier can check a signature using a randomized public key, knowing that it comes from a valid authority but not which one. Unlike solutions using ring signatures where it is the signer (credential issuer) who chooses the ring size, we let users do it independently (relying on SPS-EQ and an efficient proof of correct randomization alone). Hence, once users get a credential from a valid

Scheme	$\lvert\sigma\rvert$	$\lvert\mathsf{pk}\rvert$	Sign	Verify	ChgRep	Assumptions
[KSD19]	$8\lvert\mathbb{G}_1\rvert + 9\lvert\mathbb{G}_2\rvert$	$(2+\ell)\lvert\mathbb{G}_2\rvert$	29E	11P	19P+38E	SXDH
Section 5	$9\lvert\mathbb{G}_1\rvert + 4\lvert\mathbb{G}_2\rvert$	$(2+\ell)\lvert\mathbb{G}_2\rvert$	10E	11P	19P+21E	extKerMDH, SXDH

Table 1: Signatures comparison including pairings and exponentiations.

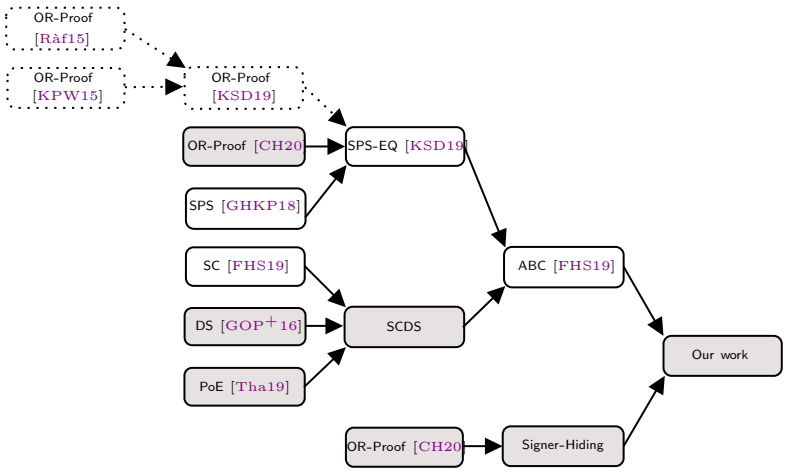

Fig. 1: Summary of building blocks used in this work. Dashed boxes represent replaced building blocks while grey boxes are used to highlight our contributions. When applicable, references inside each box indicate the related previous work.

authority they can decide on the anonymity set themselves whenever they use their credential. This approach is better aligned with the concept of self sovereign identity and related applications that seek to empower users giving them full control on their identity.

Along the way, we also describe how to build *mercurial signatures* [CL19] with security proofs in the standard model (assuming a CRS). All the previous ones [CL19, CL21] have security proofs in the GGM. Consequently, our signature construction can also be used to build delegatable anonymous credentials [CL06, BCC+09] as well.

1.3 Roadmap

We begin by presenting related work with a focus on the development of SPS-EQ and set-commitment schemes (Section 2) followed by the required cryptographic background in Section 3. Our *first* contribution, extending the set-commitment scheme (SC) in [FHS19] to support non-membership proofs for disjoint sets (DS), is presented in Section 4. We also define here the proof of exponentiation (PoE), which can be seen as an optional *plug-in* to gain efficiency in this new set-commitment scheme (SCDS).

In Section 5 we present our SPS-EQ scheme. It uses a new malleable NIZK argument based on a recent work from Couteau and Hartmann [CH20], which we use to replace the one underlying [KSD19].

In [FHS19] the authors discuss a concurrently secure variant of their ABC based on a trapdoor commitment scheme to implement ZKPoK, assuming the existence of one-way functions and a CRS. Since our SPS-EQ makes use of a CRS, we instantiate the previous variant with it, incorporate a Pedersen commitment scheme to compute the relevant ZKPoK, and adapt the rest to our set-commitment scheme and the proof of exponentiation (*second* and *third* contributions). Thus, we dedicate Section 6 to present the resulting ABC.

Subsequently, we extend the previous construction to support another NIZK argument that allows to hide the identity of the signer during showings. This allows us to build another ABC as our *fourth* contribution. Furthermore, we also outline in this section how to perform revocation and build mercurial signatures.

In Figure 1 we summarize the dependencies between the different building blocks used in the previously mentioned sections highlighting our contributions.

Finally, a detailed comparison on the concrete efficiency of our constructions when compared to other state-of-the-art alternatives is provided in Section 8, while the conclusions of this work are presented in Section 9.

2 Background and Related Work

2.1 Structure-Preserving Signatures on Equivalence Classes

In [HS14], Hanser and Slamanig introduced a novel structure preserving signature (SPS) scheme that allowed joint randomization of messages and their corresponding signatures, coining Structure-Preserving Signatures on Equivalence Classes (SPS-EQ). They observed that if one considers a prime-order group $\mathbb{G}$ and defines the projective vector space $(\mathbb{G}^*)^\ell$, there is a partition into equivalence classes given by the following relation $\mathcal{R}$: $\mathbf{m} \in (\mathbb{G}^*)^\ell \sim_{\mathcal{R}} \mathbf{m}^* \in (\mathbb{G}^*)^\ell \iff \exists\, \mu \in \mathbb{Z}_p^* : \mathbf{m}^* = \mu\mathbf{m}$. If the discrete logarithm problem is hard in $\mathbb{G}$ and one restricts the vector components to be non-zero, given two vectors $\mathbf{m}$ and $\mathbf{m}^*$, it is difficult to distinguish whether they were randomly sampled or if they belong to the same equivalence class. Hence, Hanser and Slamanig defined SPS-EQ as SPS that produce signatures on an equivalence class instead of messages alone. Given a message and its corresponding signature, SPS-EQ provides a *controlled form of malleability* in which one can publicly (without requiring access to the secret key) adapt a signature to change the representative (message). The equivalence relation provides indistinguishability on the message space if the DDH assumption holds. If additionally, updated signatures are distributed like fresh signatures, message-signature pairs falling into the same class are unlinkable. For unlinkability to hold, signatures should also be randomized when adapting them to a new representative of the class. As described in [FHS19], given a representative and its corresponding signature, a random representative of the same class with an adapted signature are indistinguishable from a random message-signature pair.

Since their introduction, SPS-EQ have been used to build several cryptographic protocols (*e.g.,* [BHKS18, FHS15, FHKS16, BHSB19, DS18, BLL+19, FGKO17]). They have been used in anonymous credentials [HS14, DHS15a, FHS19], and delegatable anonymous credential systems, in this case under the name of mercurial signatures [CL19, CL21], which are an extension of the equivalence classes to the signing keys. State-of-the-art constructions focus on building schemes under weaker assumptions and with tight security. The first step was the work from Fuchsbauer and Gay [FG18]. Subsequently, Khalili *et al.* [KSD19] proposed a new SPS-EQ which is, to the best of our knowledge, the only one under standard assumptions and with a tight security reduction to date.

The construction of [FG18] is based on the family of Matrix-Diffie-Hellman assumptions [EHK+13]. They first modify an affine MAC from [BKP14] to obtain a linear structure-preserving MAC, which is made publicly verifiable using a known technique in the context of SPS [KPW15]. This allows to use a *tag* to randomize both the signature and message. The resulting scheme is secure under a weaker notion of unforgeability (EUF-CoMA). In [KSD19], authors observe that using a structure-preserving MAC such as the one from [FG18] has an inherent problem in the security game. As messages and Matrix Decision Diffie-Hellman challenges belong to the same source group of the bilinear group, one cannot do better than EUF-CoMA security following this approach. Consequently, they proposed to use an OR-Proof based on that in [GHKP18] to then construct tightly secure structure-preserving MACs based on the key encapsulation mechanism of Gay *et al.* in [GHK17]. This allows to circumvent the previous issue and obtain the first EUF-CMA secure SPS-EQ scheme with a tight security reduction under standard assumptions.

In this work, we present an SPS-EQ scheme where the OR-based proof in [KSD19] is replaced by the one in [CH20], while adapting other building blocks accordingly.

2.2 Accumulators and Set-Commitments.

In [DHS15b], Derler, Hanser and Slamanig revisited the notion of cryptographic accumulators and proposed a unified formal model which included the notions of undeniability and indistinguishability for accumulators, complementing the classical ones of correctness and collision-freeness. They showed how to construct a commitment scheme using an indistinguishable accumulator in a black-box manner. The relation stems from the fact that indistinguishability and collision-freeness of accumulators resemble those of hiding and binding for commitments.

In subsequent work [HS14], Hanser and Slamanig built an ABC with constant-size credentials and constant-size showings (for selective disclosure of attributes) based on a polynomial commitment scheme with factor openings. They departed from the work of Kate *et al.* on constant-size polynomial commitments [KZG10] with the following observations; (1) If a credential is seen as a set of attributes mapped to roots of a monic polynomial, then one can generate a polynomial commitment of constant-size to represent the credential using the approach from [KZG10]. (2) Instead of evaluating the

Scheme	[CL04]	[CL11]	[CL13]	[CDHK15] & [FHS19]	[TG20]	[San20]	[HP20]	Section 6
	Issuing n-attr. credential							
Comm.	$O(n)$	$O(n)$	$O(n)$	$\boldsymbol{O(1)}$	$O(n)$	$\boldsymbol{O(1)}$	$O(n)$	$\boldsymbol{O(1)}$
User	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Issuer	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	Showing k-of-n attributes (selective disclosure)							
|ek|	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$
Comm.	$O(n)$	$O(1)$	$O(k)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
User	$O(n)$	$O(n)$	$O(k)$	$O(n-k)$	$O(n-k)$	$O(n-k)$	$\boldsymbol{O(1)}$	$O(\max\{n-k,k\})$
Verifier	$O(n)$	$O(n)$	$O(k)$	$O(k)$	$O(k)$	$O(k)$	$O(n)$	$\boldsymbol{O(1)}$

Table 2: Asymptotic complexities of ABC systems where n is the number of attributes in the credential and k the number of disclosed ones during a showing.

polynomial at certain points, what is important to prove possession of an attribute is to open factors of the polynomial instead. (3) If one can open multiple factors in constant-size, a showing involving a selective disclosure of attributes can be done in constant-size as well.
As a result they proposed an indistinguishable bilinear accumulator ([Ngu05]) with batch membership proofs (*i.e,* factor opening), which was subsequently re-stated as a *set-commitment* scheme in a follow-up work [FHS19].

A drawback of the ABC from [FHS19] is that the achieved level of *expressiveness* is limited. It allows only to show proofs for the conjunction of attributes in arbitrary subsets of attributes encoded in the credential (selective disclosure). Another potential issue is that verification involves a number of exponentiations that are linear in the size of the subset to be verified. This is undesirable when verification of the credential should be fast.

Thakur [Tha19] proposed a series of protocols for batch membership and non-membership proofs for bilinear accumulators using *proofs of exponentiation* (an idea previously introduced for accumulators in groups of unknown order by Boneh et. al [BBF19] and by Wesolowski [Wes20]) to shift the computational cost from the verifier to the prover. The main idea is to replace some of the exponentiations by a single polynomial division and to use of a non-interactive proof obtained via the Fiat-Shamir transform.

Batch proofs in the bilinear accumulator setting can be traced back to the works by Papamanthou *et al.* [PTT11] and by Ghosh *et al.* [GOP+16]. The latter presents the same underlying ideas of the (non)membership proofs provided by Thakur, and a *Zero-Knowledge Dynamic Universal Accumulator*, which strengthens the notion of indistinguishability using the randomization ideas from [DHS15b].

More recently, a new set-commitment scheme including set intersection and set difference operations was proposed in [TG20]. It provides more expressiveness when compared to the one from [FHS19] but under a weaker hiding notion.

We incorporate the previous ideas from [DHS15b, GOP+16], and [Tha19] to extend the set-commitment scheme from [FHS19] to support disjoint sets (batch non-membership proofs), while also allowing a faster verification and a stronger hiding notion. Thus, we obtain a set-commitment scheme that is more expressive than the one in [FHS19] and almost as expressive as [TG20] (but better in efficiency and strength).

2.3 Attribute-based Credentials

We recall in Table 2 the asymptotic complexities for the issuing and showing protocols, considering recent credential systems from each of the lines of work mentioned in the introduction, and our construction in Section 6 . For showing protocols we consider the selective disclosure of attributes (*i.e.,* the ability to show multiple attributes while hiding others during a showing). While the work from [HP20] (based on aggregatable signatures) is the only one with $O(1)$ complexity for the user during a showing, this is obtained at the cost of a more expensive verifier. Our work achieves $O(1)$ complexity for the verifier but keeping better asymptotics for the user. A more detailed comparison on the concrete efficiency of ABC's (as well as an implementation benchmark) was provided in [TG20], but the recent works from [San20] and [HP20] were not included. Therefore, we provide an updated comparison for the most efficient ones in Section 8.

2.4 Signer-Hiding

Independent and concurrent work by Bobolz *et al.* [BEK+21] also addressed the problem of hiding the identity of a credential issuer/signer under the notion of *issuer-hiding*. There, the authors propose a sligthly different setting to avoid using an OR-like proof as done in this work. In brief, the authors consider access policies of the form $\{\sigma_i, \mathsf{pk}_i\}_{i\in[n]}$, where σ_i is a signature on a given authority's public key pk_i produced by the verifier. As a result, users can prove the correspondence between a public key (defined in the policy) and the credential verification under that public key in zero knowledge, using a NIZK independent to the number of public keys defined in the policy. In this regard, we note that our work is compatible with their formalization and, furthermore, under the previous setting such NIZK can be avoided in our case. Since we use mercurial signatures, it would suffice to randomize the access policy and the user credential consistently.

3 Preliminaries

Notation. Let BGGen be a p.p.t algorithm that on input 1^λ with λ the security parameter, returns a description $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ of an asymmetric bilinear group where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p with $\lceil \log_2 p \rceil = \lambda$, P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate) bilinear map. BG is said to be of Type-3 if no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known. For all $a \in \mathbb{Z}_p$, we denote by $[a]_s = aP_s \in \mathbb{G}_s$ the implicit representation of a in $\mathbb{G}_s$ for $s \in \{1, 2\}$. For matrices (or vectors) $\mathbf{A}$, $\mathbf{B}$ we extend the pairing notation to $e([\mathbf{A}]_1, [\mathbf{B}]_2) := [\mathbf{AB}]_T \in \mathbb{G}_T$. Sampling r from set $\mathcal{S}$ uniformly at random is denoted by $r \xleftarrow{\$} \mathcal{S}$. Finally, we use the notation $\mathcal{A}(x; y)$ to indicate that a value y (usually computed internally by $\mathcal{A}$), is being passed directly to $\mathcal{A}$ on input x.

Assumptions. We recall in the full version (Appendix A) the Diffie-Hellman assumptions in the billinear group setting and the algebraic framework from [EHK+13] and [MRV16], including a generalization of the Strong Diffie-Hellman assumption from [FHS19]. Besides, we will also use the following generalization of the KerMDH assumption introduced in [CH20]. It allows an adversary to extend the given matrix but requiring it to output multiple, linearly independent vectors in the kernel.

$\mathcal{D}_k$-$\mathsf{extKerMDH}$Assumption. Let $\mathcal{D}_k$ be a matrix distribution, $l, k \in \mathbb{N}$, and $s \in \{1, 2\}$. We say that the $\mathcal{D}_k$-$\mathsf{extKerMDH}$ *assumption* holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{D} \xleftarrow{\$} \mathcal{D}_k$, and all p.p.t. adversaries $\mathcal{A}$ the following probability is negligible.

$$\Pr\left[\begin{array}{c|c} [\mathbf{C}]_{3-s} \in \mathbb{G}_3^{l+1\times k+l+1} \wedge [\mathbf{B}]_s \in \mathbb{G}_s^{l\times k} & \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); \mathbf{D} \xleftarrow{\$} \mathcal{D}_k \\ \wedge\ [\mathbf{C}]_{3-s}[\mathbf{D}']_s = 0 & ([\mathbf{C}]_{3-s}, [\mathbf{B}]_s) \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, [\mathbf{D}]_s) \\ \wedge\ \mathsf{rank}(\mathbf{C}) \geq l+1 & [\mathbf{D}']_s := [\frac{\mathbf{D}}{\mathbf{B}}]_s \end{array}\right]$$

Characteristic Polynomial. For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$, we refer to $\mathsf{Ch}_\mathcal{X}(X) = \prod_{x\in\mathcal{X}}(X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$ (a monic polynomial of degree $n = |\mathcal{X}|$ and defined over $\mathbb{Z}_p[X]$) as its *characteristic polynomial*. For a group generator P, $\mathsf{Ch}_\mathcal{X}(s)P$ can be efficiently computed (*e.g.,* using the Fast Fourier Transform) when given $(s^i P)_{i=0}^{|\mathcal{X}|}$ but not s. This is because $\mathsf{Ch}_\mathcal{X}(s)P = \sum_{i=0}^{i=n}(c_i \cdot s^i)P$.

In addition to exploiting properties of characteristic polynomials, we will also use the Schwartz-Zippel lemma and the Extended Euclidean Algorithm (EEA) in our constructions following the ideas from [GOP+16].

Lemma 1 (Schwartz-Zippel). Let $q_1(x), q_2(x)$ be two d-degree polynomials from $\mathbb{Z}_p[X]$ with $q_1(x) \neq q_2(x)$, then for $s \xleftarrow{\$} \mathbb{Z}_p$, the probability that $q_1(s) = q_2(s)$ is at most d/p, and the equality can be tested in time $O(d)$.

3.1 Non-interactive Zero-Knowledge Arguments and Malleable Proof Systems

We next define fully adaptive NIZK arguments (*i.e.,* the crs does not depend on the language distribution or language parameters), and the notions of malleable proof systems given in [CKLM12] and [KSD19] respectively.

NIZK Syntax. A fully adaptive NIZK Π for a family of language distribution $\{\mathcal{D}_{\mathsf{pp}}\}_{\mathsf{pp}}$ consists of four probabilistic algorithms:

$\mathsf{PGen}(1^\lambda)$: On input 1^λ generates public parameters pp, a crs and a trapdoor td.
$\mathsf{Prove}(\mathsf{crs}, \rho, x, w)$: On input a crs, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$ and a statement x with witness w, outputs a proof π for $x \in \mathcal{L}_\rho$.
$\mathsf{Verify}(\mathsf{crs}, \rho, x, \pi)$: On input a crs, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$, a statement x and a proof π, accepts or rejects the proof.
$\mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, \rho, x)$: Given a crs, the trapdoor td, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$ and a statement x, outputs a simulated proof for the statement $x \in \mathcal{L}_\rho$.

The following properties need to hold for NIZK arguments with respect to a family of language distributions $\mathcal{D}_{\mathsf{pp}}$.

Perfect Completeness.

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \rho, x, \pi) = 1 \;\middle|\; \begin{array}{c}(\mathsf{pp}, \mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda); \rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}}); \\ (x, w) \in R_\rho; \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, \rho, x, w)\end{array}\right] = 1$$

Computational Soundness. For every efficient adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{crs}, \rho, x, \pi) = 1 \\ \wedge\ x \notin \mathcal{L}_\rho\end{array} \;\middle|\; \begin{array}{c}(\mathsf{pp}, \mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda); \\ \rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}}); (\pi, x) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \rho)\end{array}\right] \approx 0$$

where the probability is taken over PGen.

Perfect Zero-Knowledge. For all λ, all $(\mathsf{pp}, \mathsf{crs}, \mathsf{td}) \in \mathsf{Supp}(\mathsf{PGen}(1^\lambda))$, all $\rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}})$ and all $(x, w) \in R_\rho$, the distributions $\mathsf{Prove}(\mathsf{crs}, \rho, x, w)$ and $\mathsf{SimProve}(\mathsf{crs}, \mathsf{td}, \rho, x)$ are identical.

Let $\mathcal{R}_\mathcal{L}$ be the witness relation associated to a language $\mathcal{L}$, then a controlled malleable proof system is accompanied by a family of efficiently computable n-ary transformations $T = (T_x, T_w)$ such that for any n-tuple $\{(x_1, w_1), \ldots, (x_n, w_n)\} \in \mathcal{R}_\mathcal{L}^n$ it holds that $(T_x(x_1, \ldots, x_n), T_w(w_1, \ldots, w_n)) \in \mathcal{R}_\mathcal{L}$. Intuitively, such a proof system allows when given valid proofs $\{\Omega_i\}_{i\in[n]}$ for words $\{x_i\}_{i\in[n]}$ with associated witnesses $\{w_i\}_{i\in[n]}$ to publicly compute a valid proof Ω for word $x := T_x(x_1, \ldots, x_n)$ corresponding to witness $w := T_w(w_1, \ldots, w_n)$ using an additional algorithm ZKEval which is defined as follows:

$\mathsf{ZKEval}(\mathsf{crs}, \mathcal{T}, (x_i, \Omega_i)_{i\in[n]})$ takes as input a common reference string crs, a transformation $T \in \mathcal{T}$, words $x_1, \ldots, x_n$ and their corresponding proofs $\Omega_1, \ldots, \Omega_n$, and outputs a new word $x' := T_x(x_1, \ldots, x_n)$ and proof Ω'.

Proofs computed by ZKEval should be indistinguishable from freshly computed proofs for the resulting word x' and corresponding witness w'. This notion is captured by the following definition.

Derivation Privacy. A NIZK proof system Π, malleable with respect to a set of transformations $\mathcal{T}$ defined on some relation $\mathcal{R}$ is *derivation private*, if for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\Pr\left[\begin{array}{l}\mathsf{crs} \xleftarrow{\$} \mathsf{PGen}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ (\mathsf{st}, ((x_i, w_i), \Omega_i)_{i\in[q]}, T) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}), \\ \textbf{if } (T \notin \mathcal{T} \vee (\exists\ i \in [q] : (\mathsf{Verify}(\mathsf{crs}, x_i, \Omega_i) = 0) \vee (x_i, w_i) \notin \mathcal{R}) \\ \textbf{return } \bot, \\ \textbf{else if } b = 0 : \Omega \leftarrow \mathsf{Prove}(\mathsf{crs}, T_x((x_i)_{i\in[q]}), T_w((w_i)_{i\in[q]})), \\ \textbf{else if } b = 1 : \Omega \leftarrow \mathsf{ZKEval}(\mathsf{crs}, T, (x_i, \pi_i)_{i\in[q]}), \\ b' \xleftarrow{\$} \mathcal{A}(\mathsf{st}, \Omega)\end{array} : b = b'\right]$$

4 A Set-Commitment Scheme supporting Disjoint Sets

We extend the set-commitment scheme in [FHS19] to support *non-membership proofs* for disjoint sets, while also including an optional *proof of exponentiation* to replace most of the exponentiations in the verifier (outsourcing them to the prover) with a single polynomial division. To do so, we borrow the previously mentioned ideas in [DHS15b], [GOP+16] and [Tha19], and adapt them to the Type-3 setting.

SCDS Syntax. A *set-commitment scheme supporting disjoint sets* (SCDS) consists of the following p.p.t algorithms:

$\mathsf{Setup}(1^\lambda, 1^q)$ is a probabilistic algorithm which takes as input a security parameter λ and an upper bound q for the cardinality of committed sets, both in unary form. It outputs public parameters pp (including an evaluation key ek), and discards the trapdoor key s used to generate them. $\mathbb{Z}_p^* \setminus \{s\}$ defines the domain of set elements for sets of maximum cardinality q.

$\mathsf{TSetup}(1^\lambda, 1^q)$ is equivalent to Setup but also returns the trapdoor key.

$\mathsf{Commit}(\mathsf{pp}, \mathcal{X})$ is a probabilistic algorithm which takes as input pp and a set $\mathcal{X}$ with $1 \leq |\mathcal{X}| \leq q$. It outputs a commitment C on set $\mathcal{X}$ and opening information O.

$\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O)$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, and opening information O. It outputs 1 if and only if O is a valid opening of C on $\mathcal{X}$.

$\mathsf{OpenSS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{S})$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, opening information O, and a non-empty set $\mathcal{S}$. If $\mathcal{S}$ is a subset of $\mathcal{X}$ committed to in C, OpenSS outputs a witness wit that attests to it. Otherwise, outputs $\perp$.

$\mathsf{OpenDS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, opening information O, and a non-empty set $\mathcal{D}$. If $\mathcal{D}$ is disjoint from $\mathcal{X}$ committed to in C, OpenDS outputs a witness $\underline{\mathsf{wit}}$ that attests to it. Otherwise, outputs $\perp$.

$\mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wit})$ is a deterministic algorithm which takes as input pp, a commitment C, a non-empty set $\mathcal{S}$, and a witness wit. If wit is a valid witness for $\mathcal{S}$ a subset of the set committed to in C, it outputs 1 and otherwise $\perp$.

$\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \underline{\mathsf{wit}})$ takes as input pp, a commitment C, a non-empty set $\mathcal{D}$, and a witness wit. If $\underline{\mathsf{wit}}$ is a valid witness for $\mathcal{D}$ being disjoint from the set committed to in C, it outputs 1 and otherwise $\perp$.

$\mathsf{PoE}(\mathsf{pp}, \mathcal{X}, \alpha)$ takes as input pp, a non-empty set $\mathcal{X}$, and a randomly-chosen value α. It computes a proof of exponentiation for the characteristic polynomial of $\mathcal{X}$ and outputs a proof π_Q and a witness Q.

A SCDS scheme is *secure* if it satisfies the properties of correctness, binding, hiding, and soundness. These notions are defined next, modified to suit the scheme, but following the usual convention.

Correctness. An SCDS scheme is *correct* if for all $q > 0$, all $\lambda > 0$, all $\mathsf{pp} \in [\mathsf{Setup}(1^\lambda, 1^q)]$, all non-empty $\mathcal{S} \subseteq \mathcal{X}$ and all non-empty $\mathcal{D} : \mathcal{D} \cap \mathcal{X} = \emptyset$, the following probabilities equal 1:

1. $\mathsf{Pr}\left[(C, O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, \mathcal{X}) \ : \ \mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1\right]$
2. $\mathsf{Pr}\left[\begin{array}{l}(C, O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, \mathcal{X}); \\ \mathsf{wit} \leftarrow \mathsf{OpenSS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{S})\end{array} : \mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wit}) = 1\right]$
3. $\mathsf{Pr}\left[\begin{array}{l}(C, O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, \mathcal{X}); \\ \underline{\mathsf{wit}} \leftarrow \mathsf{OpenDS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})\end{array} : \mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \underline{\mathsf{wit}}) = 1\right]$

Binding. An SCDS scheme is *binding* if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\mathsf{Pr}\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q), \\ (C, \mathcal{X}, O, \mathcal{X}', O') \xleftarrow{\$} \mathcal{A}(\mathsf{pp})\end{array} : \begin{array}{r}\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1 \ \wedge \\ \mathsf{Open}(\mathsf{pp}, C, \mathcal{X}', O') = 1 \ \wedge \ \mathcal{X} \neq \mathcal{X}'\end{array}\right]$$

Hiding. We say that an SCDS scheme is *hiding* if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$ with access to $\mathcal{O}_{\mathsf{SS}}$, an opening oracle which allows queries for sets $\mathcal{X}' \subseteq \mathcal{X}_0 \cap \mathcal{X}_1$, and to $\mathcal{O}_{\mathsf{DS}}$, for sets $\mathcal{X}'$ s.t. $\mathcal{X}' \cap \{\mathcal{X}_0 \cup \mathcal{X}_1\} = \emptyset$, there is a negligible function $\epsilon(\cdot)$ such that:

$$\mathsf{Pr}\left[\begin{array}{l}b \xleftarrow{\$} \{0,1\}; \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathcal{X}_0, \mathcal{X}_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}); \\ (C, O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, \mathcal{X}_b); \\ b^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{SS}}(\mathsf{pp}, C, \mathcal{X}_b, O, \cdot), \mathcal{O}_{\mathsf{DS}}(\mathsf{pp}, C, \mathcal{X}_b, O, \cdot)}(\mathsf{st}, C)\end{array} : b^* = b\right] - \frac{1}{2} \leq \epsilon(k).$$

where $\mathcal{X}_0$ and $\mathcal{X}_1$ are two distinct sets s.t. $1 \leq |\mathcal{X}_b| \leq q$.
If the above holds for $\epsilon \equiv 0$, the scheme is said to be perfectly hiding.

Soundness. An SCDS scheme is sound if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$, the following probabilities are negligible,

1. $\mathsf{Pr}\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (C, \mathcal{X}, O, \mathcal{S}, \mathsf{wit}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp})\end{array} : \begin{array}{r}\mathcal{S} \nsubseteq \mathcal{X} \ \wedge \ \mathsf{OpenSS}(C, \mathcal{X}, O) = 1 \\ \wedge \ \mathsf{VerifySS}(C, \mathcal{S}, \mathsf{wit}) = 1\end{array}\right]$
2. $\mathsf{Pr}\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (C, \mathcal{X}, O, \mathcal{D}, \underline{\mathsf{wit}}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp})\end{array} : \begin{array}{r}\mathcal{D} \cap \mathcal{X} \neq \emptyset \ \wedge \ \mathsf{OpenDS}(C, \mathcal{X}, O) = 1 \\ \wedge \ \mathsf{VerifyDS}(C, \mathcal{D}, \underline{\mathsf{wit}}) = 1\end{array}\right]$

$\underline{\mathsf{SCDS.Setup}}(1^\lambda, 1^q)$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{pp} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i\in[q]})$
return pp

$\underline{\mathsf{SCDS.TSetup}}(1^\lambda, 1^q)$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{pp} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i\in[q]})$
return (pp, s)

$\underline{\mathsf{SCDS.PoE}}(\mathsf{pp}, \mathcal{X}, \alpha)$:
$Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$; Let $h(X)$ and β s.t.
$\mathsf{Ch}_{\mathcal{X}}(X) = (X+\alpha)\cdot h(X) + \beta$; $\pi_Q \leftarrow h(s)P_2$
return (π_Q, Q)

$\underline{\mathsf{SCDS.Commit}}(\mathsf{pp}, \mathcal{X})$:
if $|\mathcal{X}| > q$ **return** $\perp$; $r \xleftarrow{\$} \mathbb{Z}_p^*$
if $\exists\ s' \in \mathcal{X} : s'P_1 = sP_1$
  $C \leftarrow rP_1$; $O \leftarrow (1, (r, s'))$
else $C \leftarrow r \cdot \mathsf{Ch}_{\mathcal{X}}(s)P_1$; $O \leftarrow (0, r)$
return (C, O)

$\underline{\mathsf{SCDS.Open}}(\mathsf{pp}, C, \mathcal{X}, O)$:
if $O = (1, (r, s')) \wedge\ s'P_1 = sP_1$
  if $C = rP_1$ **return** 1 **else** 0
if $O = (0, r)$
  if $C = r \cdot \mathsf{Ch}_{\mathcal{X}}(s)P_1$ **return** 1 **else** 0

$\underline{\mathsf{SCDS.OpenSS}}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{S})$:
if $\mathsf{SCDS.Open}(C, \mathcal{X}, O) = 0 \vee$
  $\mathcal{S} \nsubseteq \mathcal{X} \vee \mathcal{S} = \emptyset$ **return** $\perp$
if $O = (1, (r, s'))$
  if $s' \notin \mathcal{S}$ **return** $\mathsf{Ch}_{\mathcal{S}}(s')^{-1} C$
if $O = (0, r)$ **return** $r \cdot \mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
else return $\perp$

$\underline{\mathsf{SCDS.VerifySS}}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wit}, [\mathsf{PoE}])$:
if $(\mathcal{S} = \emptyset \wedge \mathsf{wit} = \perp)$ **return** 1
if $\exists\ s' \in \mathcal{S} : s'P_1 = sP_1$
  if $\mathsf{wit} = \perp$ **return** 1 **else** 0
if $\mathsf{PoE} = \perp$
  return $e(\mathsf{wit}, \mathsf{Ch}_{\mathcal{S}}(s)P_2) = e(C, P_2)$
else
  parse $\mathsf{PoE} = (\alpha, \pi_Q, Q)$
  $\beta \leftarrow \mathsf{Ch}_{\mathcal{S}}(X) (\mathrm{mod}\ (X+\alpha))$
  return $e(sP_1 + \alpha P_1, \pi_Q) + e(\beta P_1, P_2)$
  $= e(P_1, Q) \wedge e(\mathsf{wit}, Q) = e(C, P_2)$

$\underline{\mathsf{SCDS.OpenDS}}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})$:
if $(t = 0 \vee |\mathcal{D} \cap \mathcal{X}| > 0)$ **return** $\perp$
if $O = (1, (r, s'))$
  if $s' \in \mathcal{D}$ **return** $\perp$ **else**
  $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$; $(w_0, w_1) \leftarrow (\gamma P_2, \frac{1-\gamma\cdot r}{\mathsf{Ch}_{\mathcal{D}}(s)} P_1)$
if $O = (0, r)$
  $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$; Let $q_1(X)$ and $q_2(X)$ s.t.
  $\mathsf{Ch}_{\mathcal{X}}(X)\cdot q_1(X) + \mathsf{Ch}_{\mathcal{D}}(X)\cdot q_2(X) = 1$
  $q_1'(s) \leftarrow q_1(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{D}}(s)$
  $q_2'(s) \leftarrow q_2(s) - \gamma \cdot \mathsf{Ch}_{\mathcal{X}}(s)$
  $(w_0, w_1) \leftarrow ((r^{-1} \cdot q_1'(s))P_2, q_2'(s)P_1)$
return (w_0, w_1)

$\underline{\mathsf{SCDS.VerifyDS}}(\mathsf{pp}, C, \mathcal{D}, \underline{\mathsf{wit}}, [\mathsf{PoE}])$:
if $(\mathcal{D} = \emptyset \wedge \underline{\mathsf{wit}} = \perp)$ **return** 1
if $\exists\ s' \in \mathcal{D} : s'P_1 = sP_1$
  if $\underline{\mathsf{wit}} = \perp$ **return** 1 **else** 0
parse $\underline{\mathsf{wit}} = (w_0, w_1)$
if $\mathsf{PoE} = \perp$ **return**
  $e(C, w_0) + e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) = e(P_1, P_2)$
else
  parse $\mathsf{PoE} = (\alpha, \pi_Q, Q)$
  $\beta \leftarrow \mathsf{Ch}_{\mathcal{D}}(X) (\mathrm{mod}\ (X+\alpha))$
  return $e(sP_1 + \alpha P_1, \pi_Q) + e(\beta P_1, P_2)$
  $= e(P_1, Q) \wedge e(C, w_0) + e(w_1, Q) = e(P_1, P_2)$

Fig. 2: Our SCDS construction

4.1 Construction

Our construction is presented in Figure 2. As in [FHS19] we use a special opening for the case in which the commited set contains the trapdoor to achieve perfect correctness and perfect hiding. To prove that a given set is disjoint with respect to the commited set, the EEA is computed to obtain the Bézout coefficients. This way, equality is checked randomizing q_1, q_2 and using a single PPE. Finally, the PoE computes a polynomial division, and produces the corresponding proof.

Theorem 1. The SCDS construction from Figure 2 is correct and perfectly hiding. Furthermore, if the q-co-DL (resp. q-co-GSDH) assumption holds, SCDS is computationally binding (resp. sound).

Proof. The proof strategy follows closely that of [FHS19]. We extend these proofs in a similar manner to consider disjoint sets. The full proof is provided in the full version (Appendix B).

5 Our SPS-EQ construction

The starting point for the SPS-EQ construction in [KSD19] was the tightly secure SPS from [GHKP18], which builds on a structure-preserving MAC (based on the works from [GHK17] and [Hof17]) and a NIZK OR-Proof from [Ràf15]. To couple with equivalence classes, the authors proposed a way to adapt the OR-Proof so that it could be randomized and malleable. Unfortunately, as the CRS used in the OR-Proof from [Ràf15] was incompatible with the required randomization properties, the authors were forced to build a QA-NIZK on top to overcome the limitation.

In this section we introduce a new SPS-EQ scheme based on the one from [KSD19], which we obtain replacing the underlying OR-Proof from [Ràf15] with one given in [CH20], while adapting accordingly. As a result we obtain a more efficient signature scheme based on a new malleable OR-NIZK argument. Before giving the intuition of our construction, we recall the syntax and security properties for SPS-EQ introduced in [FHS19] and [KSD19].

SPS-EQ Syntax. An SPS-EQ consists of the following p.p.t algorithms:

$\mathsf{ParGen}(1^\lambda)$ is a probabilistic algorithm which takes as input a security parameter λ and returns public parameters pp including an asymmetric bilinear group, but without the related trapdoor.

$\mathsf{TParGen}(1^\lambda)$ is like the ParGen algorithm but it also returns the trapdoor.

$\mathsf{KGen}(\mathsf{pp}, \ell)$ is a probabilistic algorithm which takes as input pp and a vector length $\ell > 1$, and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathbf{m})$ is a probabilistic algorithm which takes as input pp, a representative $\mathbf{m} \in (\mathbb{G}_i^*)^\ell$ for class $[\mathbf{m}]_\mathcal{R}$, a secret key sk, and outputs a signature $\sigma' = (\sigma, \tau)$ (potentially including a tag τ) on the message $\mathbf{m}$.

$\mathsf{ChgRep}(\mathsf{pp}, \mathbf{m}, (\sigma, \tau), \mu, \mathsf{pk})$ is a probabilistic algorithm which takes as input pp, a representative message $\mathbf{m} \in (\mathbb{G}_i^*)^\ell$, a signature σ (and potentially a tag τ), a scalar μ and a public key pk. It computes an updated signature σ' on new representative $\mathbf{m}^* = \mu\mathbf{m}$ and returns $(\mathbf{m}^*, \sigma')$.

$\mathsf{Verify}(\mathsf{pp}, \mathbf{m}, (\sigma, \tau), \mathsf{pk})$ is a deterministic algorithm which takes as input pp, a representative message $\mathbf{m}$, a signature σ (potentially including a tag τ) and public key pk. If σ is a valid signature on $\mathbf{m}$ it outputs 1 and 0 otherwise.

Correctness. An SPS-EQ scheme over $(\mathbb{G}_i^*)^\ell$ is correct if for any $\lambda \in \mathbb{N}$, any $\ell > 1$, any $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$, any pair $(\mathsf{sk}, \mathsf{pk})$, any message $\mathbf{m} \in (\mathbb{G}_i^*)^\ell$, and any $\mu \in \mathbb{Z}_p^*$, the following holds:

$$\Pr\left[\mathsf{Verify}(\mathbf{m}, \mathsf{Sign}(\mathsf{sk}, \mathbf{m}), \mathsf{pk}) = 1\right] = 1, \text{ and}$$
$$\Pr\left[\mathsf{Verify}(\mathsf{ChgRep}(\mathbf{m}, \mathsf{Sign}(\mathsf{sk}, \mathbf{m}), \mu, \mathsf{pk}), \mathsf{pk}) = 1\right] = 1.$$

EUF-CMA. An SPS-EQ scheme over $(\mathbb{G}_i^*)^\ell$ is existentially unforgeable under adaptively chosen-message attacks, if for all $\ell > 1$ and p.p.t adversaries $\mathcal{A}$ with access to a signing oracle Sign, the following probability is negligible,

$$\Pr\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda),\\ (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell),\\ ([\mathbf{m}]_i^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\textsc{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk})\end{array} : \begin{array}{r}[\mathbf{m}^*]_\mathcal{R} \neq [\mathbf{m}]_\mathcal{R} \ \forall\ [\mathbf{m}]_i \in \mathsf{Q} \ \wedge\\ \mathsf{Verify}([\mathbf{m}]_i^*, \sigma^*, \mathsf{pk}) = 1\end{array}\right],$$

where Q is the set of queries that $\mathcal{A}$ has issued to the signing oracle Sign. Note that in the tag-based case this oracle returns (σ_i, τ_i).

The following notion is based on Definition 10 from [KSD19], which defines perfect adaption of signatures in the CRS model. Perfect adaption mandates that signatures output by the algorithm ChgRep are distributed identically to new signatures on the respective representative. When this notion is defined considering adversaries who could maliciously generate signing keys, one obtains the strongest possible notion for perfect adaption. Unlike [KSD19], we opt to explicitly state that perfect adaption is defined with respect to the message space. We do this, as later on we will introduce a new a definition for perfect adaption with respect to the *key space*.

Perfect adaption of signatures (under malicious keys in the honest parameters model) with respect to the *message space*: An SPS-EQ over $\mathcal{S}_\mathbf{m}$ perfectly adapts signatures with respect to the message space if for all tuples $(\mathsf{pp}, \mathsf{pk}, [\mathbf{m}]_i, \sigma, \mu)$ where $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$, $[\mathbf{m}]_i \in \mathcal{S}_\mathbf{m}$, $\mu \in \mathbb{Z}_p^*$, and $\mathsf{Verify}([\mathbf{m}]_i, \sigma, \mathsf{pk}) = 1$, we have that the output of $\mathsf{ChgRep}([\mathbf{m}]_i, (\sigma, \tau), \mu, \mathsf{pk})$ is $([\mu \cdot \mathbf{m}]_i, \sigma^*)$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}([\mu \cdot \mathbf{m}]_i, \sigma^*, \mathsf{pk}) = 1$.

5.1 Our Malleable NIZK argument

Our malleable NIZK argument is based solely on the fully-adaptive OR-Proof from [CH20]. This allows us to circumvent the randomization problem in the OR-Proof from [Ràf15], and to avoid the need to build a QA-NIZK atop.

As a result, we reduce the number of exponentiations required in the proving and ZKEval algorithms, which leads to a more efficient signature scheme. This comes at the cost of relying on the

$\underline{\mathsf{PGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $z \xleftarrow{\$} \mathbb{Z}_p$
return $((\mathsf{BG}, [z]_2), z)$

$\underline{\mathsf{PPro}(\mathsf{crs}, [\mathbf{x}_1]_1, \mathbf{w}_1, [\mathbf{x}_2]_1, \mathbf{w}_2)}$:
// $[\mathbf{x}_j]_1 = \mathbf{A}_i\mathbf{w}_j$ with $\mathbf{A} \in \mathcal{M}^{2k\times k}$
$\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$; $z_{1-i} \xleftarrow{\$} \mathbb{Z}_p^*$; $\delta \xleftarrow{\$} \mathbb{Z}_p^*$
$[z_i]_2 \leftarrow \delta[z]_2 - [z_{1-i}]_2$
$[\mathbf{d}_i^j]_2 \leftarrow [z_i]_2\mathbf{w}_j + [\mathbf{s}_j]_2$
$[\mathbf{a}_i^j]_1 \leftarrow [\mathbf{A}_i]\mathbf{s}_j$
$\mathbf{d}_{1-i}^j \xleftarrow{\$} \mathbb{Z}_p^k$
$[\mathbf{a}_{1-i}^j]_1 \leftarrow \mathbf{A}_{1-i}\mathbf{d}_{1-i}^j - z_{1-i}\mathbf{x}_j$
return $([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, \delta P_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$

$\underline{\mathsf{PSim}(\mathsf{crs}, z, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1)}$:
$z_0 \xleftarrow{\$} \mathbb{Z}_p$; $\delta \xleftarrow{\$} \mathbb{Z}_p^*$; $z_1 \leftarrow \delta z - z_0$
for all $i \in \{0,1\}$, $j \in \{1,2\}$ **do**
  $\mathbf{d}_i^j \xleftarrow{\$} \mathbb{Z}_p^k$; $[\mathbf{a}_i^j]_1 \leftarrow \mathbf{A}_i\mathbf{d}_i^j - z_i\mathbf{x}_j$
return $([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, \delta P_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$

$\underline{\mathsf{PRVer}(\mathsf{crs}, [\mathbf{x}]_1, \pi)}$:
parse $\pi = ([\mathbf{a}_i]_1, [\mathbf{d}_i]_2, [z_i]_2, Z_1)_{i\in\{0,1\}}$
check $e(Z_1, [z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$
for all $i \in \{0,1\}$ **check**
  $e([\mathbf{A}_i]_1, [\mathbf{d}_i]_2) = e([\mathbf{x}]_1, [z_i]_2) + e([\mathbf{a}_i]_1, [1]_2)$

$\underline{\mathsf{PVer}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega)}$:
parse $\Omega = ([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, Z_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$
check $e(Z_1, [z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$
for all $i \in \{0,1\}$, $j \in \{1,2\}$ **check**
  $e([\mathbf{A}_i]_1, [\mathbf{d}_i^j]_2) = e([\mathbf{x}_j]_1, [z_i]_2) + e([\mathbf{a}_i^j]_1, [1]_2)$

$\underline{\mathsf{ZKEval}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega)}$:
parse $\Omega = ([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, Z_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$
check $\mathsf{PVer}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega)$
$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$; $Z_1' \leftarrow \alpha Z_1$
for all $i \in \{0,1\}$
  $[z_i']_2 \leftarrow \alpha[z_i]_2$; $[\mathbf{a}_i']_1 \leftarrow \alpha[\mathbf{a}_i^1]_1 + \alpha\beta[\mathbf{a}_i^2]_1$
  $[\mathbf{d}_i']_2 \leftarrow \alpha[\mathbf{d}_i^1]_2 + \alpha\beta[\mathbf{d}_i^2]_2$
return $([\mathbf{a}_i']_1, [\mathbf{d}_i']_2, [z_i']_2, Z_1')$

Fig. 3: Malleable NIZK argument for language $\mathcal{L}^\vee_{\mathbf{A}_0,\mathbf{A}_1}$

$\mathcal{L}_1$-1-extKerMDH assumption. We argue that the change is justified as the extKerMDH is a natural extension of the KerMDH assumption and in this case, the assumption is also falsifiable.

Intuition. We look for a NIZK proof which can be randomizable and malleable so that randomized proofs look like fresh proofs, while the malleability allows to update the proof statements. The goal is to obtain derivation privacy, which is crucial to perform the change of representative in the signature scheme.

The fully-adaptive NIZK argument from [CH20] is based on a challenge $z = z_0 + z_1$, where z is in the CRS, and z_0 and z_1 are elements of the proof and chosen such that the equation holds. To randomize a proof we need to randomize z_0 and z_1 and so, instead of checking the original equation we will check for linear combinations of the equation $\alpha z = z_0 + z_1$. We modify the original proof to compute a random α and add an extra element $Z = \alpha P_1$ to the proof. Consequently, the verification algorithm will now check an extra pairing.

As observed in [KSD19], the malleability of the OR-NIZK proof can be achieved by using a tag and a second NIZK for that tag with shared randomness. We follow the same approach. The resulting malleable NIZK argument for the OR-language (for fixed $\mathbf{A}_0$ and $\mathbf{A}_1$) is defined below and presented in Figure 3.

$$\mathcal{L}^\vee_{\mathbf{A}_0,\mathbf{A}_1} = \{[\mathbf{x}]_1 \in \mathbb{G}_1^{2k} | \exists\ \mathbf{w} \in \mathbb{Z}_p^k : [\mathbf{x}]_1 = [\mathbf{A}_0]_1 \cdot \mathbf{w} \vee [\mathbf{x}]_1 = [\mathbf{A}_1]_1 \cdot \mathbf{w}\},$$

Theorem 2. The protocol in Figure 3 is a fully adaptive NIZK argument for the OR-language $\mathcal{L}^\vee_{\mathbf{A}_0,\mathbf{A}_1}$ if the falsifiable $\mathcal{L}_1$-$(4k+1)$-extKerMDH assumption holds in $\mathbb{G}_2$.

Proof. The proof follows [CH20] and is provided in the full version (Appendix C).

5.2 Signature Construction

Our construction is shown in Figure 4, where the highlighted sections note the main differences to the scheme presented in [KSD19] In the full version (Appendix H), we also show how to extend it to obtain mercurial signatures (later explained in Section 7.1).

Theorem 3. The SPS-EQ in Figure 4 perfectly adapts signatures (under malicious keys in the honest parameter model) with respect to the message space.

To prove Theorem 3 we follow almost verbatim the original proof from [KSD19].

Proof. For all $[\mathbf{m}]_1$ and $\mathsf{pk} = ([\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$, a signature $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ generated according to the CRS $([\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [z]_2)$ satisfying the verification algorithm must be of the

SPS-EQ.ParGen(1^λ):
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
return $(\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$

SPS-EQ.TParGen(1^λ):
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$
return $(\mathsf{pp}, \mathsf{td})$

SPS-EQ.KGen($\mathsf{pp}, 1^\lambda$):
$\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$; $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times 2}$
$[\mathbf{B}]_2 \leftarrow [\mathbf{K}_0]_2[\mathbf{A}]_2$; $[\mathbf{C}]_2 \leftarrow [\mathbf{K}]_2[\mathbf{A}]_2$
$\mathsf{sk} \leftarrow (\mathbf{K}_0, \mathbf{K})$; $\mathsf{pk} \leftarrow ([\mathbf{B}]_2, [\mathbf{C}]_2)$
return $(\mathsf{sk}, \mathsf{pk})$

SPS-EQ.Sign($\mathsf{pp}, \mathsf{sk}, [\mathbf{m}]_1$):
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1$; $[\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
$\Omega \leftarrow \mathsf{PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
parse $\Omega = (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
$\mathbf{u}_1 \leftarrow \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\mathbf{m}]_1$; $\mathbf{u}_2 \leftarrow \mathbf{K}_0^\top[\mathbf{w}]_1$
$\sigma \leftarrow ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
$\tau \leftarrow ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$
return (σ, τ)

SPS-EQ.Verify($\mathsf{pp}, [\mathbf{m}]_1, (\sigma, \tau), \mathsf{pk}$):
parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \bot\}$
check $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) = e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$
if $\tau \neq \bot$ **check**
  $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
  $e([\mathbf{u}_2]_1^\top, [\mathbf{A}_2]) = e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$

SPS-EQ.ChgRep($\mathsf{pp}, [\mathbf{m}]_1, \sigma, \tau, \mu, \mathsf{pk}$):
parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \bot\}$
$\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
check $\mathsf{PVer}(\mathsf{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$
check $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$
$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$
$[\mathbf{u}_1']_1 \leftarrow \mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1$
$[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$
for all $i \in \{0, 1\}$
  $[z_i']_2 \leftarrow \alpha[z_i]_2$
  $[\mathbf{a}_i']_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$
  $[d_i']_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$
$\Omega' \leftarrow ((([\mathbf{a}_i']_1, [d_i']_2, [z_i']_2)_{i\in\{0,1\}}, \alpha Z_1)$
$\sigma' \leftarrow ([\mathbf{u}_1']_1, [\mathbf{t}']_1, \Omega')$
return $(\mu[\mathbf{m}]_1, \sigma')$

Fig. 4: Our SPS-EQ scheme.

form: $\sigma = (\mathbf{K}_0^\top[\mathbf{A}_0]_1 r_1 + \mathbf{K}^\top[\mathbf{m}]_1, [\mathbf{A}_0]_1 r_1, [\mathbf{A}_0]s_1, [\mathbf{A}_1]d_1^1 - z_1[\mathbf{A}_0]_1 r_1, [z_0]_2 r_1 + [s_1]_2, [d_1^1]_2, [z_0]_2, [z_1]_2, Z_1)$. A signature output by ChgRep has the form $\sigma = (\mathbf{K}_0^\top[\mathbf{A}_0]_1(\mu r_1 + \beta r_2) + \mathbf{K}^\top[\mu\mathbf{m}]_1, [\mathbf{A}_0]_1(\mu r_1 + \beta r_2), [\mathbf{A}_0]\alpha(\mu s_1 + \beta s_2), [\mathbf{A}_1]\alpha(\mu d_1^1 + \beta d_1^2) - z_1[\mathbf{A}_0]_1\alpha(\mu r_1 + \beta r_2), \alpha([z_0]_2(\mu r_1 + \beta r_2) + \mu[s_1]_2 + \beta[s_2]_2), \alpha(\mu[d_1^1]_2 + \beta[d_1^2]_2), \alpha[z_0]_2, \alpha[z_1]_2, \alpha Z_1)$, for new independent randomness α, β and μ so is a random element in the space of all signatures. Furthermore, the signature output by ChgRep is distributed identically to a fresh signature on message $[\mathbf{m}]_1$ output by Sign. □

Theorem 4. If the KerMDH and MDDH assumptions hold, the SPS-EQ in Figure 4 is unforgeable.

Proof. The proof is provided in the full version (Appendix D).

6 Extending the ABC Model from [FHS19]

In this section we present a new ABC model which extends [FHS19] to consider NAND showing proofs and the use of a CRS (denoted as pp). A NAND showing proof allows users to demonstrate that a given set of attributes is not present in their credential. The core differences in this extended ABC model follow naturally from (1) the addition of disjoint sets in the SCDS scheme in section 4, and (2) the removal of the key verification algorithm (as we work with a CRS).

ABC Syntax. An ABC scheme consists of the following p.p.t algorithms:

Setup($1^\lambda, 1^q$) takes a security parameter λ and an upper bound q for the size of attribute sets, and outputs public parameters pp discarding any trapdoor.
TSetup($1^\lambda, 1^q$) similar to Setup but it also returns a trapdoor (if any).
OrgKeyGen(pp) takes pp as input and outputs an organization key pair (osk, opk).
UserKeyGen(pp) takes pp as input and outputs a user key pair (usk, upk).
Obtain(pp, usk, opk, $\mathcal{X}$) and Issue(pp, upk, osk, $\mathcal{X}$) are run by a user and the organization respectively, who interact during execution. Obtain takes as input pp, the user's secret key usk, an organization's public key opk, and an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < t$. Issue takes as input pp, a user public key

upk, the organization's secret key osk, and an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < t$. At the end of this protocol, Obtain outputs a credential cred on $\mathcal{X}$ for the user or $\perp$ if the execution failed.

$\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathcal{X}, \mathcal{S}, \mathcal{D}, \mathsf{cred})$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})$ are run by a user and a verifier respectively, who interact during execution. Show takes as input pp, an organization public key opk, a credential cred for the attribute set $\mathcal{X}$, potentially non-empty sets $\mathcal{S} \subseteq \mathcal{X}$, $\mathcal{D} \nsubseteq \mathcal{X}$ representing attributes sets being a subset ($\mathcal{S}$) or disjoint ($\mathcal{D}$) to the attribute set ($\mathcal{X}$) committed in the credential. Verify takes as input pp, an organization public key opk, the sets $\mathcal{S}$ and $\mathcal{D}$. At the end, Verify outputs 1 or 0 indicating whether or not the credential showing was accepted.

6.1 Security Properties

The following notions are based on the security model from [FHS19] (Section 5.1), which we adapt to consider the use of a crs (pp) and NAND showing proofs. Informally, an ABC scheme is secure if it has the following properties:

Correctness. A showing of a credential with respect to a non-empty sets $\mathcal{S}$ and $\mathcal{D}$ of attributes always verify if the credential was issued honestly on some attribute set $\mathcal{X}$ with $\mathcal{S} \subset \mathcal{X}$ and $\mathcal{D} \nsubseteq \mathcal{X}$.

Unforgeablility. Given at least one non-empty set $\mathcal{S} \subset \mathcal{X}$ or $\mathcal{D} \nsubseteq \mathcal{X}$, a user in possession of a credential for the attribute set $\mathcal{X}$ cannot perform a valid showing for $\mathcal{D} \subset \mathcal{X}$ nor for $\mathcal{S} \nsubseteq \mathcal{X}$. Moreover, no coalition of malicious users can combine their credentials and prove possession of a set of attributes which no single member has. This holds even after seeing showings of arbitrary credentials by honest users (thus, covering replay attacks).

Anonymity. During a showing, no verifier and no (malicious) organization (even if they collude) is able to identify the user or learn anything about the user, except that she owns a valid credential for the shown attributes. Furthermore, different showings of the same credential are unlinkable.

To introduce the corresponding formal definitions, the following global variables and oracles are listed below.

Global variables. At the beginning of each experiment, either the experiment computes an organization key pair $(\mathsf{osk}, \mathsf{opk})$ or the adversary outputs opk. In the anonymity game there is a bit b, which the adversary must guess.

In order to keep track of all honest and corrupt users, we introduce the sets $\mathtt{HU}$, and $\mathtt{CU}$, respectively. We use the lists $\mathtt{UPK}$, $\mathtt{USK}$, $\mathtt{CRED}$, $\mathtt{ATTR}$ and $\mathtt{OWNR}$ to track user public and secret keys, issued credentials and corresponding attributes and to which user they were issued. Furthermore, we use the sets J_{LoR} and I_{LoR} to store which issuance indices and corresponding users have been set during the first call to the left-or-right oracle in the anonymity game.

Oracles. Considering an adversary $\mathcal{A}$ the oracles are as follows:

$\mathcal{O}_{\mathtt{HU}}(i)$ takes as input a user identity i. If $i \in \mathtt{HU} \cup \mathtt{CU}$, it returns $\perp$. Otherwise, it creates a new honest user i by running $(\mathtt{USK}[i], \mathtt{UPK}[i]) \xleftarrow{\$} \mathsf{UsrKGen}(\mathsf{opk})$, adding i to the honest user list $\mathtt{HU}$ and returning $\mathtt{UPK}[i]$.

$\mathcal{O}_{\mathtt{CU}}(i, \mathsf{upk})$ takes as input a user identity i and (optionally) a user public key upk; if user i does not exist, a new corrupt user with public key upk is registered, while if i is honest, its secret key and all credentials are leaked. In particular, if $i \in \mathtt{CU}$ or if $i \in I_{\mathsf{LoR}}$ (that is, i is a challenge user in the anonymity game) then the oracle returns $\perp$. If $i \in \mathtt{HU}$ then the oracle removes i from $\mathtt{HU}$ and adds it to $\mathtt{CU}$; it returns $\mathtt{USK}[i]$ and $\mathtt{CRED}[j]$ for all j with $\mathtt{OWNR}[j] = i$. Otherwise (*i.e.,* $i \notin \mathtt{HU} \cup \mathtt{CU}$), it adds i to $\mathtt{CU}$ and sets $\mathtt{UPK}[i] \leftarrow \mathsf{upk}$.

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{X})$ takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \mathtt{HU}$, it returns $\perp$. Otherwise, it issues a credential to i by running

$$(\mathsf{cred}, \top) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \mathtt{USK}[i], \mathsf{opk}, \mathcal{X}), \mathsf{Issue}(\mathsf{pp}, \mathtt{UPK}[i], \mathsf{osk}, \mathcal{X}).$$

If $\mathsf{cred} = \perp$, it returns $\perp$. Else, it appends $(i, \mathsf{cred}, \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious organization, issue a credential to an honest user. It takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \mathtt{HU}$, it returns $\perp$. Otherwise, it runs

$$(\mathsf{cred}, \cdot) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \mathtt{USK}[i], \mathsf{opk}, \mathcal{X}), \cdot),$$

where the Issue part is executed by $\mathcal{A}$. If $\mathsf{cred} = \bot$, it returns $\bot$. Else, it appends $(i, \mathsf{cred}, \mathcal{X})$ to $(\texttt{OWNR}, \texttt{CRED}, \texttt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious user, obtain a credential from an honest organization. It takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \texttt{CU}$, it returns $\bot$. Otherwise, it runs

$$(\cdot, I) \xleftarrow{\$} (\cdot, \mathsf{Issue}(\mathsf{pp}, \texttt{UPK}[i], \mathsf{osk}, \mathcal{X})),$$

where the Obtain part is executed by $\mathcal{A}$. If $\mathrm{I} = \bot$, it returns $\bot$. Else, it appends $(i, \bot, \mathcal{X})$ to $(\texttt{OWNR}, \texttt{CRED}, \texttt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Show}}(j, \mathcal{S}, \mathcal{D})$ lets the adversary $\mathcal{A}$ play a dishonest verifier during a showing by an honest user. It takes as input an index of an issuance j and attributes sets $\mathcal{S}$ and $\mathcal{D}$. Let $i \xleftarrow{\$} \texttt{OWNR}[j]$. If $i \notin \texttt{HU}$, it returns $\bot$. Otherwise, it runs

$$(S, \cdot) \xleftarrow{\$} \mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \texttt{ATTR}[j], \mathcal{S}, \mathcal{D}, \texttt{CRED}[j]), \cdot)$$

where the Verify part is executed by $\mathcal{A}$.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S}, \mathcal{D})$ is the challenge oracle in the anonymity game where $\mathcal{A}$ must distinguish (multiple) showings of two credentials $\texttt{CRED}[j_0]$ and $\texttt{CRED}[j_1]$. The oracle takes two issuance indices j_0 and j_1 and attribute sets $\mathcal{S}$ and $\mathcal{D}$. If $J_{\mathsf{LoR}} \neq \emptyset$ and $J_{\mathsf{LoR}} \neq \{j_0, j_1\}$, it returns $\bot$. Let $i_0 \xleftarrow{\$} \texttt{OWNR}[j_0]$ and $i_1 \xleftarrow{\$} \texttt{OWNR}[j_1]$. If $J_{\mathsf{LoR}} \neq \emptyset$ then it sets $J_{\mathsf{LoR}} \xleftarrow{\$} \{j_0, j_1\}$ and $I_{\mathsf{LoR}} \xleftarrow{\$} \{i_0, i_1\}$. If $i_0, i_1 \neq \texttt{HU} \vee \mathcal{S} \nsubseteq \texttt{ATTR}[j_0] \cap \texttt{ATTR}[j_1] \vee \mathcal{D} \cap \{\texttt{ATTR}[j_0] \cup \texttt{ATTR}[j_1]\} \neq \emptyset$, it returns $\bot$. Else, it runs

$$(S, \cdot) \xleftarrow{\$} (\mathsf{Show}(\mathsf{opk}, \texttt{ATTR}[j_b], \mathcal{S}, \mathcal{D}, \texttt{CRED}[j_b]), \cdot),$$

(with b set by the experiment) where the Verify part is executed by $\mathcal{A}$.

Correctness. An ABC system is correct, if for all $\lambda > 0$, all $t > 0$, all $\mathcal{X}$ with $0 < |\mathcal{X}| \leq t$ and all $\emptyset \neq \mathcal{S} \subset \mathcal{X}$ and $\emptyset \neq \mathcal{D} \nsubseteq \mathcal{X}$ with $0 < |\mathcal{D}| \leq t$ it holds that:

$$\mathsf{Pr}\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UsrKGen}(\mathsf{pp}); \\ (\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X}), \\ \qquad\qquad \mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})) \end{array} : \begin{array}{l} (\top, 1) \xleftarrow{\$} (\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathcal{X}, \mathcal{S}, \\ \mathcal{D}, \mathsf{cred}), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})) \end{array}\right] = 1.$$

Unforgeability. An ABC system is unforgeable, if for all $\lambda > 0$, all $q > 0$ and p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\texttt{HU}}, \mathcal{O}_{\texttt{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}\}$ the following probability is negligible.

$$\mathsf{Pr}\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{pp}); \\ (\mathcal{S}, \mathcal{D}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{opk}); \\ (\cdot, b^*) \xleftarrow{\$} (\mathcal{A}(\mathsf{st}), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})) \end{array} : \begin{array}{l} b^* = 1 \wedge \\ \forall\, j : \texttt{OWNR}[j] \in \texttt{CU} \implies \\ (\mathcal{S} \notin \texttt{ATTR}[j] \vee \mathcal{D} \in \texttt{ATTR}[j]) \end{array}\right]$$

Anonymity. An ABC system is anonymous, if for all $\lambda > 0$, all $q > 0$ and all p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\texttt{HU}}, \mathcal{O}_{\texttt{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{LoR}}\}$ the following probability is negligible.

$$\mathsf{Pr}\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); b \xleftarrow{\$} \{0,1\}; (\mathsf{opk}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}); \\ b^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{st}) \end{array} : b^* = b\right] - \tfrac{1}{2}$$

7 Our ABC construction

As previously explained in Section 1.3, our ABC scheme is based on the one from [FHS19]. The main changes are the following:

– As we use a signature scheme that relies on a CRS, we move the parameters of the set-commitment scheme from the organization's key pair to the public parameters pp that include the previous CRS. Furthermore, we instantiate the ZKPoK's using Pedersen commitments and the construction from [Dam00], as suggested in [FHS19] (Remark 1).

– Our showing protocol can be instantiated with two sets $\mathcal{S}$ and $\mathcal{D}$, one to compute AND proofs (selective disclosure) and one to compute NAND proofs.
– We integrate the proof of exponentiation to the showing protocol [5].

Intuition. We begin explaining the difference to [FHS19] with respect to malicious organizations as it clarifies the changes introduced in the issuing protocol. We recall that in this context the term *malicious organizations* refers to organizations whose key-pairs are generated in a way that trapdoor information is included. Such trapdoor information could later be used by an organization to break anonymity, provided that extra information (a transcript of a given showing protocol containing a credential issued by the organization) is available. The ABC scheme from [FHS19] defines a ZKPoK in the issuing protocol ($\Pi^{\mathcal{R}_O}$) for which the organization needs to prove knowledge of the corresponding secret key to avoid the previous scenario. Since the signing keys in our SPS-EQ need to be generated using the CRS (which includes the matrix $\mathbf{A}$), we do not need to request a ZKPoK from the organization in the issuing protocol as the signature's verification algorithm a pairing involving the matrix $\mathbf{A}$ and the organization's public key $\mathsf{opk} = (\mathbf{B}, \mathbf{C})$ is used to check the signature. Hence, a signature that verifies rules out that 1) someone impersonated the issuer signing with a different secret key, and 2) that the public key was maliciously generated. Regarding the showing protocol, the only changes are the addition of NAND and exponentiation proofs. For the latter, we require the verifier to randomly pick the challenge and send it to the user.

For ease of exposition, we present the resulting construction (Scheme 1) in Figure 5 considering selective disclosures only. We highlight in grey the required changes to do NAND proofs, but both types of proofs could be computed while executing a single showing. If so, a NAND proof increases bandwidth by 4 elements (two from $\mathbb{G}_1$ and two from $\mathbb{G}_2$), as the PoE can reuse the same challenge.

Theorem 5. Scheme 1 is correct.

Theorem 6. If the q-co-DL assumption holds, the ZKPoK's have perfect ZK, SCDS is sound, and SPS-EQ is EUF-CMA secure, then Scheme 1 is unforgeable.

Theorem 7. If the DDH assumption holds, the ZKPoK's have perfect ZK, and the SPS-EQ perfectly adapts signatures, then Scheme 1 is anonymous.

Proof. Proof of Theorem 6 follows closely to that presented in [FHS19] but extended to include disjoint sets. Proof of Theorem 7 also follows that in [FHS19] with the exception that we work with a CRS and an accordingly modified definition of perfect adaption. All proofs are provided in the full version (Appendix E).

7.1 Revocation strategies

The natural approach to revocation would be to follow that described in [DHS15a] where they use the fact that randomization of a credential is compatible with the randomization of the accumulator and its corresponding witness. This approach requires the revocation authority to compute and maintain the witness list. As it uses the accumulator from [ATSM09], the cost of non-membership proofs is linear in the size of the accumulator (*i.e.,* revoked users), and this should be done at least once by the manager for every user. If, instead, the dynamic variant is used (as discussed in [DHS15a]), then users could be given their non-membership witness once and subsequently update it with a single constant size operation. Other approaches for revocation are discussed in the full version (Appendix F).

7.2 Signer-Hiding

We recall that our signature scheme is based on the one from [KSD19] and that we are using the credential framework of [FHS19]. Therefore, as we have $k = 1$ and $\ell = 3$, the public keys consist of two vectors $[\mathbf{B}]_2 \in (\mathbb{G}_2^*)^2$ and $[\mathbf{C}]_2 \in (\mathbb{G}_2^*)^3$, where the secret keys have the form $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K})$ with $\mathbf{K}_0 \xleftarrow{\$} (\mathbb{Z}_p^*)^{2\times 2}$ and $\mathbf{K} \xleftarrow{\$} (\mathbb{Z}_p^*)^{3\times 2}$. With this in mind, we can naturally define equivalence relationships on the key spaces $\mathcal{S}_{\mathsf{sk}} = \{(\mathbb{Z}_p^*)^{2\times 2} \times (\mathbb{Z}_p^*)^{3\times 2}\}$ and $\mathcal{S}_{\mathsf{pk}} = \{(\mathbb{G}_2^*)^2 \times (\mathbb{G}_2^*)^3\}$ as follows:

[5] The security of this integration is discussed in the full version (Appendix J).

$\underline{\mathsf{ABC.Setup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$; $(\mathsf{sps}_{\mathsf{pp}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ParGen}(1^\lambda; \mathsf{BG})$;
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$; **return** $(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{pp}}, \mathsf{ck})$

$\underline{\mathsf{ABC.TSetup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{scds}_{\mathsf{td}}) \xleftarrow{\$} \mathsf{SCDS.TSetup}(1^\lambda, q)$; $(\mathsf{sps}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{td}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.TParGen}(1^\lambda; \mathsf{BG})$;
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$; $\mathsf{ck}_{\mathsf{td}} \leftarrow r$; **return** $((\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{pp}}, \mathsf{ck}), (\mathsf{scds}_{\mathsf{td}}, \mathsf{sps}_{\mathsf{td}}, \mathsf{ck}_{\mathsf{td}}))$

$\underline{\mathsf{ABC.OrgKGen}(\mathsf{pp})}$:
return $\mathsf{SPS\text{-}EQ.KGen}(\mathsf{BG}, \mathsf{sps}_{\mathsf{pp}}, 3)$

$\underline{\mathsf{ABC.UsrKGen}(\mathsf{pp})}$:
$\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{upk} \leftarrow \mathsf{usk}P_1$
return $(\mathsf{usk}, \mathsf{upk})$

$\mathsf{ABC.Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X})$		$\mathsf{ABC.Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$; $a \leftarrow r_1 P_1$		
$c \leftarrow \mathsf{Commit}(\mathsf{ck}, a, r_2)$	$\xrightarrow{c}$	
$z \leftarrow r_1 + e \cdot \mathsf{usk}$	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathbb{Z}_p^*$
$(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{X}; \mathsf{usk})$	$\xrightarrow{C,\ R,\ z,a,r_2}$	**if** $(zP_1 \neq a + e \cdot \mathsf{upk} \vee$
$r_3 \xleftarrow{\$} \mathbb{Z}_p^*$; $R \leftarrow r_3 C$		$c \neq \mathsf{Commit}(\mathsf{ck}, a, r_2))$ **return** $\perp$
		if $(e(C, P_2) \neq e(\mathsf{upk}, \mathsf{Ch}_{\mathcal{X}}(s) P_2)$
		$\wedge\ \forall\ x \in \mathcal{X} : xP_1 \neq \mathsf{ek}_1^0)$ **return** $\perp$
	$\xleftarrow{(\sigma,\tau)}$	$(\sigma, \tau) \leftarrow \mathsf{SPS\text{-}EQ.Sign}((C, R, P_1), \mathsf{osk})$
check $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps}_{\mathsf{pp}}, (C, R, P_1), (\sigma, \tau), \mathsf{opk})$		
return $\mathsf{cred} = (C, (\sigma, \tau), r_3, O)$		

$\mathsf{ABC.Show}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathsf{ek}, \mathcal{S}, \mathsf{cred})$		$\mathsf{ABC.Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S})$
parse $\mathsf{cred} = (C, \sigma, r, O)$; $\mu \xleftarrow{\$} \mathbb{Z}_p^*$		
if $O = (1, (o_1, o_2))$ **then**		
$O' = (1, (\mu \cdot o_1, o_2))$ **else** $O' = \mu O$		
$\sigma' \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{sps}_{\mathsf{pp}}, (C, rC, P_1), \sigma, \tau, \mu, \mathsf{opk})$		
$(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$		
$\mathsf{cred}' \leftarrow (C_1, C_2, C_3, \sigma')$		
$\mathsf{wit} \leftarrow \mathsf{SCDS.OpenSS}(\mathsf{scds}_{\mathsf{pp}}, \mu C, \mathcal{S}, \mathsf{ek}, O')$		
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$		
$c_1 \leftarrow \mathsf{Commit}(\mathsf{ck}, a_1, r_2)$	$\xrightarrow{\mathsf{cred}',\ \mathsf{wit},\ c_1,c_2}$	**parse** $\mathsf{cred}' = (C_1, C_2, C_3, \sigma)$
$c_2 \leftarrow \mathsf{Commit}(\mathsf{ck}, a_2, r_4)$		
$\pi \leftarrow \mathsf{SCDS.PoE}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{S}, \tilde{e})$	$\xleftarrow{e,\tilde{e}}$	$e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
$z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$		
$\Omega = ((z_i, a_i, r_i)_{i \in \{1,2\}}, \pi)$	$\xrightarrow{\Omega}$	**parse** $\Omega = ((z_i, a_i, r_i)_{i \in \{1,2\}}, \pi)$
		check
		$z_1 C_1 = a_1 + eC_2$; $z_2 P_1 = a_2 + eC_3$
		$c_1 = \mathsf{Commit}(\mathsf{ck}, a_1, r_2)$
		$c_2 = \mathsf{Commit}(\mathsf{ck}, a_2, r_4)$
		$\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps}_{\mathsf{pp}}, \mathsf{cred}', \mathsf{opk})$
		$\mathsf{SCDS.VerifySS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{S}, \mathsf{wit}; \pi, \tilde{e})$

Fig. 5: Scheme 1.

$\underline{\mathsf{PGen}(1^\lambda)}$:	$\underline{\mathsf{PSim}(\mathsf{crs}, \mathsf{td}, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i))}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $z \xleftarrow{\$} \mathbb{Z}_p$; $\mathsf{crs} \leftarrow (\mathsf{BG}, [z]_1)$; $\mathsf{td} \leftarrow z$	$z_1, ..., z_{n-1} \xleftarrow{\$} \mathbb{Z}_p$
return $(\mathsf{crs}, \mathsf{td})$	$[z_n]_1 \leftarrow [\mathsf{td}]_1 - \sum_{j=1}^{j=n-1} [z_j]_1$
	for all $\mathbf{i} \in [n]$ **do**
$\underline{\mathsf{PPrv}(\mathsf{crs}, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i), \rho)}$:	$d_i^1, d_i^2 \xleftarrow{\$} \mathbb{Z}_p$
// $\mathbf{B}'_i = \mathbf{B}_i \cdot \rho \wedge \mathbf{C}'_i = \mathbf{C}_i \cdot \rho$	$[\mathbf{a}_i^1]_2 \leftarrow d_i^1 \mathbf{B}_i \text{-} z_i \mathbf{B}'_i$; $[\mathbf{a}_i^2]_2 \leftarrow d_i^2 \mathbf{C}_i \text{-} z_i \mathbf{C}'_i$
$s_1, s_2, z_1, ..., z_{n-1} \xleftarrow{\$} \mathbb{Z}_p$	**return** $((([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]})$
$[z_n]_1 \leftarrow [z]_1 - \sum_{j=1}^{j=n-1} [z_j]_1$	
$[\mathbf{a}_i^1]_2 \leftarrow s_1 \mathbf{B}_i$; $[\mathbf{a}_i^2]_2 \leftarrow s_2 \mathbf{C}_i$	$\underline{\mathsf{PVer}(\mathsf{crs}, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i), \pi)}$:
$[d_i^1]_1 \leftarrow \rho[z_i]_1 + [s_1]_1$; $[d_i^2]_1 \leftarrow \rho[z_i]_1 + [s_2]_1$	**parse** $\pi = ((([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]})$
for all $j \neq i \in [n]$ **do**	$[z_n]_1 = [z]_1 - \sum_{j=1}^{j=n-1} [z_j]_1$
$d_j^1, d_j^2 \xleftarrow{\$} \mathbb{Z}_p$	**for all** $\mathbf{i} \in [n]$ **check**
$[\mathbf{a}_j^1]_2 \leftarrow d_j^1 \mathbf{B}_j \text{-} z_j \mathbf{B}'_i$; $[\mathbf{a}_j^2]_2 \leftarrow d_j^2 \mathbf{C}_j \text{-} z_j \mathbf{C}'_i$	$e([d_i^1]_1, \mathbf{B}_i) = e([z_i]_1, \mathbf{B}'_i) + e([1]_1, [\mathbf{a}_i^1]_2)$
return $((([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]})$	$e([d_i^2]_1, \mathbf{C}_i) = e([z_i]_1, \mathbf{C}'_i) + e([1]_1, [\mathbf{a}_i^2]_2)$

Fig. 6: Fully adaptive NIZK argument for $\mathcal{L}^{\vee}_{(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}}$

$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}, \tilde{\mathsf{sk}}) \in \mathcal{S}_{\mathsf{sk}} \times \mathcal{S}_{\mathsf{sk}} \mid \exists\ \rho \in \mathbb{Z}_p^* \text{ s.t } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \tilde{\mathsf{pk}}) \in \mathcal{S}_{\mathsf{pk}} \times \mathcal{S}_{\mathsf{pk}} \mid \exists\ \rho \in \mathbb{Z}_p^* \text{ s.t } \tilde{\mathsf{pk}} = \rho \cdot \mathsf{pk}\}$$

If we have a list of public keys $(\mathbf{B}_1, \mathbf{C}_1), ..., (\mathbf{B}_n, \mathbf{C}_n)$ and define the equivalence class of each public key as before ($(\mathbf{B}'_i, \mathbf{C}'_i) = (\mathbf{B}_i, \mathbf{C}_i) \cdot \rho$), we can efficiently prove that a given public key $(\mathbf{B}'_i, \mathbf{C}'_i)$ belongs to the equivalence class of one of the public keys $(\mathbf{B}_1, \mathbf{C}_1), ..., (\mathbf{B}_n, \mathbf{C}_n)$ for some $(\mathbf{B}_i, \mathbf{C}_i)$. The idea is to use a generalized version of the OR-Proof from [CH20], and building a generalized NIZK OR-Proof for the AND statements of the two components. The new language is defined as follows (remember we use $\ell = 3$):

$$\mathcal{L}_{\bigvee(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}} = \{(\mathbf{B}'_i, \mathbf{C}'_i) \in \mathbb{G}_2^{2\times\ell} \mid \exists\ \rho \in \mathbb{Z}_p^* : \vee\ (\mathbf{B}'_i = \mathbf{B}_i \cdot \rho \wedge \mathbf{C}'_i = \mathbf{C}_i \cdot \rho)_{i\in[n]}\}$$

The resulting NIZK argument is given in Figure 6.

Theorem 8. The proof system given in Figure 6 is a fully-adaptive NIZK argument for the language $\mathcal{L}_{\bigvee(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}}$.

Proof. The proof follows from Theorem 19 in [CH20]. The only difference is that we rely on the AND composition for sigma protocols to compile the one in [CH20] using the same challenge for both proofs.

We now explain how the above NIZK can be used to hide the identity of a signer. First, we need to consider a scenario in which n-authorities can issue credentials to different sets of users. As we are in the classical setting, we also assume that every user gets a credential from one of the n-authorities and that the organization keys are certified and publicly available.

When showing a credential, the verifier needs to check the signature using the corresponding public key. The idea is to use the above NIZK proof so that a user can randomize the public key and present this randomized key to the verifier, which in turn will check the NIZK to verify that the public key is valid (*i.e.,* it belongs to the equivalence class of one of the n-authorities).

Signatures need to be adapted by the users so that they can be verified with the randomized public key. Therefore, we consider the definition of mercurial signatures [CL19], which includes algorithms $\mathsf{ConvertPK}$, $\mathsf{ConvertSK}$ and $\mathsf{ConvertSig}$, and introduce the following notion.

Perfect adaption of signatures (under malicious keys in the honest parameters model) with respect to the *key space*; An SPS-EQ over a message space $\mathcal{S}_{\mathbf{m}}$ perfectly adapts signatures with respect to the key space $\mathcal{S}_{\mathsf{pk}}$ if for all tuples $(\mathsf{pp}, [\mathsf{pk}]_j, [\mathbf{m}]_i, (\sigma, \tau), \rho)$ where $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$, $[\mathsf{pk}]_j \in \mathcal{S}_{\mathsf{pk}}$, $[\mathbf{m}]_i \in \mathcal{S}_{\mathbf{m}}$, $\mathsf{Verify}([\mathbf{m}]_i, (\sigma, \tau), [\mathsf{pk}]_j) = 1$ and $\rho \in \mathbb{Z}_p^*$, we have that the output of $\mathsf{ConvertSig}([\mathbf{m}]_i, (\sigma, \tau), \rho, [\mathsf{pk}]_j)$ is σ^*, with σ^* being a random element in the space of signatures, conditioned on $\mathsf{Verify}([\mathbf{m}]_i, \sigma^*, \mathsf{ConvertPK}([\mathsf{pk}]_j, \rho)) = 1$.

$\mathsf{ConvertSig}$ is analogous to the ChgRep algorithm, but restricted to act on the equivalence class defined by the key space. The algorithms $\mathsf{ConvertPK}$ and $\mathsf{ConvertSK}$ are just defined to abstract the computation of new representatives.

As our signature construction is compatible with the joint executions of the algorithms ChgRep and ConvertSig, we define below a general notion for perfect adaption where the ChgRep algorithm acts on all the equivalence classes.

Perfect adaption of signatures (under malicious keys in the honest parameters model); An SPS-EQ over $\mathcal{S}_{\mathbf{m}}$ perfectly adapts signatures if for all tuples $(\mathsf{pp}, [\mathsf{pk}]_j, [\mathbf{m}]_i, (\sigma, \tau), \mu, \rho)$ where $\mathsf{pp} \xleftarrow{\$} \mathsf{ParGen}(1^\lambda)$, $[\mathsf{pk}]_j \in \mathcal{S}_{\mathsf{pk}}$, $[\mathbf{m}]_i \in \mathcal{S}_{\mathbf{m}}$, $\mathsf{Verify}([\mathbf{m}]_i, (\sigma, \tau), [\mathsf{pk}]_j)=1$ and $\mu, \rho \in \mathbb{Z}_p^*$ we have that the output of $\mathsf{ChgRep}([\mathbf{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{pk}]_j)$ is $([\mu \cdot \mathbf{m}]_i, \sigma^*)$, with σ^* being a random element in the space of signatures, conditioned on $\mathsf{Verify}([\mu \cdot \mathbf{m}]_i, \sigma^*, \mathsf{ConvertPK}([\mathsf{pk}]_j, \rho))=1$.

Our construction satisfies perfect adaption under *malicious keys* in the honest parameter model with respect to the message space but not with respect to the key space. Therefore we consider perfect adaption *under honestly generated keys* next.

Theorem 9. The above extension applied to the SPS-EQ from Figure 4 perfectly adapts signatures (under honestly generated keys in the honest parameter model).

Proof. It follows from the security of SPS-EQ and the definition of perfect adaption for mercurial signatures (Appendix D and H in the full version).

We now formalize the signer-hiding notion and show that our construction satisfies it.

Signer-Hiding. An ABC system supports signer-hiding if for all $\lambda > 0$, all $q > 0$, all $n > 0$, all $t > 0$, all $\mathcal{X}$ with $0 < |\mathcal{X}| \leq t$, all $\emptyset \neq \mathcal{S} \subset \mathcal{X}$ and $\emptyset \neq \mathcal{D} \not\subseteq \mathcal{X}$ with $0 < |\mathcal{D}| \leq t$, and p.p.t adversaries $\mathcal{A}$, the following holds.

$$\mathsf{Pr}\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ \forall\ i \in [n] : (\mathsf{osk}_i, \mathsf{opk}_i) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UsrKGen}(\mathsf{pp}); j \xleftarrow{\$} [n]; \\ (\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}_j, \mathcal{X}), \mathsf{Issue}(\mathsf{upk}, \mathsf{osk}_j, \mathcal{X})); \\ j^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Show}}}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, (\mathsf{opk}_i)_{i \in [n]}) \end{array} : j^* = j \right] \leq \frac{1}{n}$$

where the oracle $\mathcal{O}_{\mathsf{Show}}$ is defined as in Section 6.

Theorem 10. If the underlying signature scheme is a SPS-EQ which perfectly adapts signatures (under honestly generated keys in the honest parameter model), the resulting ABC from Section 7.2 supports signer-hiding.

Proof. Let us first observe that the adversary can guess the bit j^* with probability $1/n$. By definition of perfect adaption, for all tuples $(\mathsf{pp}, [\mathsf{opk}]_j, [\mathbf{m}]_i, (\sigma, \tau), \mu, \rho)$ s.t $(\sigma, \tau) \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{osk}_j, [\mathbf{m}]_i)$, we have that $[\mu \cdot \mathbf{m}]_i$ and $[\rho \cdot \mathsf{opk}]_j$ are identically distributed in the message and key spaces, where $([\mu \cdot \mathbf{m}]_i, \sigma^*) \leftarrow \mathsf{ChgRep}([\mathbf{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{opk}]_j)$ and $[\rho \cdot \mathsf{opk}]_j \leftarrow \mathsf{ConvertPK}(\mathsf{opk}_j, \rho)$. Furthermore, we also have that σ^* is a random element in the space of signatures conditioned on $\mathsf{Verify}([\mu \cdot \mathbf{m}]_i, \sigma^*, [\rho \cdot \mathsf{opk}]_j) = 1$. Therefore, an adversary with access to $[\mu \cdot \mathbf{m}]_i$, σ^* and $[\rho \cdot \mathsf{opk}]_j$ can only guess the bit j^* with probability at most $1/n$. □

Integration with our ABC scheme. As our NIZK argument is fully adaptive, users can choose the size of the anonymity set (*i.e.,* the set of public keys in the OR-Proof). We find this approach much simpler than using delegatable credentials to achieve a similar result as users do not need to interact with the organizations to compute the NIZK proof nor to adapt the signature. Moreover, there is no need to use pseudonyms for public and secret keys. We essentially compute public key's pseudonyms "on-the-fly" guaranteeing that the signature adaption is done with respect to a valid public key. In other words, our NIZK argument is a proof of correct randomization, where the same randomizer is used to adapt the signature and generate a pseudonymous public key. A complete figure for the proposed ABC, including the signer-hiding extension as well as NAND proofs, is given in the full version (Appendix I).

Efficiency analysis. As the proof size is $9n - 1$ for an anonymity set of n-authorities, communication bandwidth will no longer be constant. Nevertheless, given the previously mentioned advantages we believe that this is a fair trade-off for the added functionality. In terms of computational cost, it is also substantially more efficient than similar variants (see, for instance, Table 2 from [CH20]).

ABC	[San20]	[HP20]	[TG20]	[FHS19]	Section 7
Parameters size (n-attributes)					
ek	$(\frac{n^2+n+2}{2})_{\mathbb{G}_1}+n_{\mathbb{G}_2}$	$(2n+2)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+$ $(n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+$ $(n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+$ $(n+1)_{\mathbb{G}_2}$
Cred	$2_{\mathbb{G}_2}$	$4_{\mathbb{G}_1}$	$1_{\mathbb{G}_1}+6_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+1_{\mathbb{G}_2}+2_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1}+6_{\mathbb{G}_2}+3_{\mathbb{Z}_p}$
Bandwidth					
Issue	$4_{\mathbb{G}_2}+2_{\mathbb{Z}_p}$	$n_{\mathbb{G}_1}$	$3_{\mathbb{G}_1}+(n+3)_{\mathbb{Z}_p}$	$12_{\mathbb{G}_1}+1_{\mathbb{G}_2}+8_{\mathbb{Z}_p}$	$14_{\mathbb{G}_1}+11_{\mathbb{G}_2}+7_{\mathbb{Z}_p}$
Show	$2_{\mathbb{G}_1}+2_{\mathbb{G}_2}+1_{\mathbb{G}_T}+2_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+1_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+5_{\mathbb{Z}_p}$	$10_{\mathbb{G}_1}+1_{\mathbb{G}_2}+8_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1}+14_{\mathbb{G}_2}+4_{\mathbb{Z}_p}$
k-of-n attributes (AND)					
Usr	$(2(n\text{-}k)+2)_{\mathbb{G}_1}$,$2_{\mathbb{G}_2}$, **1**	$6_{\mathbb{G}_1}$	$(6+n\text{-}k)_{\mathbb{G}_1}$	$(11+n\text{-}k)_{\mathbb{G}_1}$,$1_{\mathbb{G}_2}$, **8**	$(20+n\text{-}k)_{\mathbb{G}_1}$, $(k\text{-}1)_{\mathbb{G}_2}$,**19**
Ver	$(k+1)_{\mathbb{G}_1}$,$1_{\mathbb{G}_T}$,**5**	$4_{\mathbb{G}_1}$,$2n_{\mathbb{G}_2}$,**3**	$5_{\mathbb{G}_1}$,$(k+1)_{\mathbb{G}_2}$,**3**	$4_{\mathbb{G}_1}$,$(k+1)_{\mathbb{G}_2}$,**10**	$10_{\mathbb{G}_1}$,**16**
k-of-n attributes (NAND)					
Usr	N/A	N/A	$(6+n)_{\mathbb{G}_1}$	N/A	$(31+n)_{\mathbb{G}_1}$, $(9+2k)_{\mathbb{G}_2}$,**19**
Ver	N/A	N/A	$(2k+5)_{\mathbb{G}_1}$, $(k+3)_{\mathbb{G}_2}$,**3**	N/A	$10_{\mathbb{G}_1}$,**17**

Table 3: Efficiency of ABCs considering issuing and showing interactions (the number of pairings is marked in bold).

8 Comparison of state-of-the-art ABC

We provide comparisons on the efficiency of state-of-the-art ABC and ours (Section 7) on Table 3. For ease of exposition, we list the work from [FHS19] next to ours, and consider an instantiation of it in the CRS model, and using the same ZKPoK's as the ones used in Section 7.

When looking at a whole, the work from Sanders [San20] presents very good results while also allowing showings to prove relationships between attributes and to consider malicious keys. Nevertheless, security of the related construction is proven in the GGM model and thus, falls short in that aspect. The same also applies to the works from [HP20] and [FHS19].

While for the comparisons only the classical setting (credentials are issued by a single authority) was considered, it is worth to mention that [HP20] does consider multi-authorities. As authors point out, in order to allow multi-authorities they base their construction on aggregate signatures, and obtain the most efficient showing for the users. Their security model follows the game-based approach from [FHS19] but because of the multi-authority setting, they also consider malicious credential issuers, with adaptive corruptions, and collusions with malicious users. Unfortunately, this is done assuming that the keys are *honestly* generated.

[TG20] uses a set-commitment scheme which alongside an SDH-based signature, leads to a credential system that supports a variety of show proofs for complex statements among which AND and NAND are included. For this reason, we also compare our work with the one from [TG20] considering NAND showings. In terms of security models, authors provide a formalization for impersonation attacks and prove their scheme secure against impersonation under active and concurrent attacks. The security of their ABC scheme is proven in the standard model and providing a tight reduction.

Considering the different trade-offs, our ABC provides very similar performance when compared to [FHS19] and it is not too distant from the most efficient ones either. Unlike the rest, it can be adapted to different scenarios in case that reducing the verification cost is not needed, and it can also be efficiently adapted to provide revocation features. Furthermore, as for many practical applications the ability to perform AND and NAND showings suffices, we also achieve a good level of expressiveness too. Finally, the signer-hiding feature makes it suitable for scenarios in which the rest of the alternatives struggle.

9 Conclusions and Future Work

Our results explore multiple paths to extend the ABC framework of [FHS19] to include more applications and scenarios where it can be used. In order to improve expressiveness of the set-commitment scheme in [FHS19] we allow openings on sets of attributes disjoint from those possessed by a user. We also enhance efficiency by employing the trick of allowing the prover to compute a proof of exponentiation leaving the verifier only to compute a polynomial division.

Our signature scheme is based on [KSD19] where we adapt the SPS-EQ scheme by alleviating the need to build a QA-NIZK incorporating results from the recent framework of [CH20]. With this fully adaptive NIZK, we find further interesting applications by looking at equivalence classes on the key-space. We develop a signer-hiding notion to allow a credential-bearing user to hide their issuing organization upon presentation of the credential. As we increasingly see cases of (algorithmic) bias against users, notions such as this are of growing importance. Moreover, we also present interesting directions to integrate revocation features.

We worked in the classical setting where each credential is issued by a single authority. It would be interesting to follow the related work on aggregatable signatures to see if we could lift SPS-EQ to the multi-authority setting.

While our set-commitment scheme is more expressive than [FHS19] it is still less expressive than [TG20]. Hence, it would be interesting to see if the set-commitment scheme introduced there would yield greater expressiveness to the ABC scheme from this work. Likewise, to verify if the stronger security notions presented here, could enhance the construction in [TG20].

Acknowledgements. We thank the anonymous reviewers for their valuable feedback. The European Commission partially supported Octavio Perez Kempner's work as part of the CUREX project (H2020-SC1-FA-DTS-2018-1 under grant agreement No 826404).

References

AJO+19. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with Tighter Security. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 669–699, Cham, 2019. Springer International Publishing. Cited on page 28.

ATSM09. Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for ddh groups and their application to attribute-based anonymous credential systems. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, pages 295–308, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. Cited on pages 15 and 36.

BBF19. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 561–586, Cham, 2019. Springer International Publishing. Cited on page 5.

BCC+09. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 108–125, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. Cited on page 3.

BEK+21. Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 158–178, Cham, 2021. Springer International Publishing. Cited on page 6.

BHKS18. Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with Flexible Public Key: Introducing Equivalence Classes for Public Keys. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 405–434, Cham, 2018. Springer International Publishing. Cited on page 4.

BHSB19. Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership Privacy for Fully Dynamic Group Signatures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2181–2198, New York, NY, USA, 2019. Association for Computing Machinery. Cited on page 4.

BKP14. Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) Identity-Based Encryption from Affine Message Authentication. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 408–425, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. Cited on page 4.

BL13. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1087–1098, 11 2013. Cited on page 2.

BLL+19. Xavier Bultel, Pascal Lafourcade, Russell W. F. Lai, Giulio Malavolta, Dominique Schröder, and Sri Aravinda Krishnan Thyagarajan. Efficient Invisible and Unlinkable Sanitizable Signatures. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 159–189, Cham, 2019. Springer International Publishing. Cited on page 4.

Bra00. Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000. Cited on page 2.

CDHK15. Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and Modular Anonymous Credentials: Definitions and Practical Constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 262–288, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. Cited on pages 2 and 5.

CH20. Geoffroy Couteau and Dominik Hartmann. Shorter Non-interactive Zero-Knowledge Arguments and ZAPs for Algebraic Languages. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 768–798, Cham, 2020. Springer International Publishing. Cited on pages 3, 4, 6, 10, 11, 17, 18, 20, 26, and 27.

CKLM12. Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 281–300, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. Cited on page 6.

CL02. Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Proceedings of the 3rd International Conference on Security in Communication Networks*, SCN'02, page 268–289, Berlin, Heidelberg, 2002. Springer-Verlag. Cited on page 2.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. Cited on pages 2 and 5.

CL06. Melissa Chase and Anna Lysyanskaya. On Signatures of Knowledge. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 78–96, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. Cited on page 3.

CL11. Sébastien Canard and Roch Lescuyer. Anonymous Credentials from (Indexed) Aggregate Signatures. In *Proceedings of the 7th ACM Workshop on Digital Identity Management*, DIM '11, page 53–62, New York, NY, USA, 2011. Association for Computing Machinery. Cited on pages 2 and 5.

CL13. Sébastien Canard and Roch Lescuyer. Protecting Privacy by Sanitizing Personal Data: A New Approach to Anonymous Credentials. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, page 381–392, New York, NY, USA, 2013. Association for Computing Machinery. Cited on pages 2 and 5.

CL19. Elizabeth C. Crites and Anna Lysyanskaya. Delegatable Anonymous Credentials from Mercurial Signatures. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, pages 535–555, Cham, 2019. Springer International Publishing. Cited on pages 3, 4, 17, and 37.

CL21. Elizabeth C. Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *Proceedings on Privacy Enhancing Technologies*, 2021(4):441–463, 2021. Cited on pages 3, 4, and 37.

CLPK22. Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 409–438, Cham, 2022. Springer International Publishing. Cited on page 1.

Dam00. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 418–430, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. Cited on page 14.

DHS15a. David Derler, Christian Hanser, and Daniel Slamanig. A New Approach to Efficient Revocable Attribute-Based Anonymous Credentials. In Jens Groth, editor, *Cryptography and Coding*, pages 57–74, Cham, 2015. Springer International Publishing. Cited on pages 2, 4, 15, and 36.

DHS15b. David Derler, Christian Hanser, and Daniel Slamanig. Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives. In Kaisa Nyberg, editor, *Topics in Cryptology — CT-RSA 2015*, pages 127–144, Cham, 2015. Springer International Publishing. Cited on pages 4, 5, and 7.

DS18. David Derler and Daniel Slamanig. Highly-Efficient Fully-Anonymous Dynamic Group Signatures. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, page 551–565, New York, NY, USA, 2018. Association for Computing Machinery. Cited on page 4.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An Algebraic Framework for Diffie-Hellman Assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 129–147, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. Cited on pages 4, 6, and 24.

FG18. Georg Fuchsbauer and Romain Gay. Weakly Secure Equivalence-Class Signatures from Standard Assumptions. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 153–183, Cham, 2018. Springer International Publishing. Cited on page 4.

FGKO17. Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access Control Encryption for Equality, Comparison, and More. In Serge Fehr, editor, *Public-Key Cryptography – PKC 2017*, pages 88–118, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg. Cited on page 4.

FHKS16. Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical Round-Optimal Blind Signatures in the Standard Model from Weaker Assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 391–408, Cham, 2016. Springer International Publishing. Cited on pages 2 and 4.

FHS15. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical Round-Optimal Blind Signatures in the Standard Model. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 233–253, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. Cited on page 4.

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials. *Journal of Cryptology*, 32:498–546, 2019. Cited on pages 2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 14, 15, 19, 20, 24, 25, 26, 32, 34, 36, and 37.

GHK17. Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-Desmedt Meets Tight Security. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 133–160, Cham, 2017. Springer International Publishing. Cited on pages 4 and 9.

GHKP18. Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More Efficient (Almost) Tightly Secure Structure-Preserving Signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 230–258, Cham, 2018. Springer International Publishing. Cited on pages 3, 4, 9, 24, and 28.

GOP+16. Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-Knowledge Accumulators and Set Algebra. In *Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032*, page 67–100, Berlin, Heidelberg, 2016. Springer-Verlag. Cited on pages 3, 5, 6, and 7.

Hof17. Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 489–518, Cham, 2017. Springer International Publishing. Cited on page 9.

HP20. Chloé Hébant and David Pointcheval. Traceable Constant-Size Multi-Authority Credentials. Cryptology ePrint Archive, Report 2020/657, 2020. Cited on pages 2, 5, and 19.

HS14. Christian Hanser and Daniel Slamanig. Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 491–511, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. Cited on page 4.

KB21. Ioanna Karantaidou and Foteini Baldimtsi. Efficient constructions of pairing based accumulators. Cryptology ePrint Archive, Report 2021/638, 2021. https://eprint.iacr.org/2021/638. Cited on pages 36 and 37.

KPW15. Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-Preserving Signatures from Standard Assumptions, Revisited. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 275–295, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. Cited on pages 3 and 4.

KSD19. Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-Preserving Signatures on Equivalence Classes from Standard Assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 63–93, Cham, 2019. Springer International Publishing. Cited on pages 2, 3, 4, 6, 9, 10, 11, 15, 20, 26, 27, 28, and 37.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 177–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. Cited on page 4.

MRV16. Paz Morillo, Carla Ràfols, and Jorge L. Villar. The Kernel Matrix Diffie-Hellman Assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 729–758, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. Cited on page 6.

Ngu05. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 275–292, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. Cited on page 5.

Ped92. Torben Pryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. Cited on page 25.

PTT11. Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal verification of operations on dynamic sets. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 91–110, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. Cited on page 5.

Ràf15. Carla Ràfols. Stretching groth-sahai: Nizk proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 247–276, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. Cited on pages 3, 9, and 10.

SABB+19. Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers. In *The Network and Distributed System Security Symposium (NDSS)*, 2019. Cited on page 2.

San20. Olivier Sanders. Efficient Redactable Signature and Application to Anonymous Credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography – PKC 2020*, pages 628–656, Cham, 2020. Springer International Publishing. Cited on pages 2, 5, and 19.

Sho97. Victor Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, page 256–266, Berlin, Heidelberg, 1997. Springer-Verlag. Cited on page 2.

TG20. Syh-Yuan Tan and Thomas Groß. MoniPoly - An Expressive q-SDH-Based Anonymous Attribute-Based Credential System. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III*, volume 12493, pages 498–526. Springer, 2020. Cited on pages 2, 5, 19, 20, and 37.

Tha19. S. Thakur. Batching non-membership proofs with bilinear accumulators. *IACR Cryptol. ePrint Arch.*, 2019:1147, 2019. Cited on pages 3, 5, 7, and 40.

Wes20. Benjamin Wesolowski. Efficient Verifiable Delay Functions (extended version). *Journal of Cryptology*, September 2020. Cited on page 5.

Zur13. IBM Research Zurich. Specification of the identity mixer cryptographic library v2.3.0., 2013. Cited on page 2.

A Auxiliary Definitions

DL Assumption. Let BGGen be a bilinear-group generator that outputs $\mathsf{BG} = (p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P_1,P_2,e)$. The *discrete logarithm assumption* holds in $\mathbb{G}_i$ for BGGen if for all probabilistic polynomial-time (p.p.t) adversaries $\mathcal{A}$, the following probability is negligible.

$$\mathsf{Pr}\left[\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); a \xleftarrow{\$} \mathbb{Z}_p; a' \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, aP_i) \ : \ a' = a\right]$$

q-co-DL Assumption. Let BGGen be a bilinear-group generator that outputs BG=$(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P_1,P_2,e)$. The *q-co-discrete logarithm assumption* holds in BGGen if for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible.

$$\mathsf{Pr}\left[\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); a \xleftarrow{\$} \mathbb{Z}_p; a' \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, (a^j P_1, a^j P_2)_{j\in[q]}) \ : \ a' = a\right]$$

DDH Assumption. Let BGGen be a bilinear-group generator that outputs BG=$(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P_1,P_2,e)$. The *decisional Diffie-Hellman assumption* holds in $\mathbb{G}_i$ for BGGen, if for all p.p.t adversaries $\mathcal{A}$ the following probability is negligible.

$$\mathsf{Pr}\left[\begin{matrix} b \xleftarrow{\$} \{0,1\}, \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda), r,s,t \xleftarrow{\$} \mathbb{Z}_p \\ b^* \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, rP_i, sP_i, ((1-b)\cdot t + b\cdot rs)P_i) \end{matrix} \ : \ b^* = b\right] - \frac{1}{2}$$

SXDH Assumption. Let BGGen be a bilinear-group generator that outputs BG=$(p,\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T,P_1,P_2,e)$. The *symmetric external Diffie-Hellman assumption* holds for BGGen if DDH holds in $\mathbb{G}_1$ and in $\mathbb{G}_2$.

q-co-Generalized SDH Assumption. Let BGGen be a bilinear-group generator that outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. The *q-co-generalized-strong-Diffie-Hellman assumption* (introduced in [FHS19]) holds for BGGen, if for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible.

$$\mathsf{Pr}\left[\begin{matrix} \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda), s \xleftarrow{\$} \mathbb{Z}_p^* \\ (Q, f_1, f_2) \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, (s^i P_1, s^i P_2)_{0\le i\le q}) \end{matrix} \ : \ \begin{matrix} Q \in \mathbb{G}_1 \ \wedge f_1, f_2 \in \mathbb{Z}_p[X] \ \wedge \\ 0 \le deg\ f_1 < deg\ f_2 \le q \ \wedge \\ e(Q, f_2(s)P_2) = e(f_1(s)P_1, P_2) \end{matrix}\right]$$

Matrix Distribution. Let $k \in N$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.

$\mathcal{D}_k$-MDDH Assumption. Let $\mathcal{D}_k$ be a matrix distribution. We say that the $\mathcal{D}_k$-*Matrix Diffie-Hellman assumption* holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and all p.p.t adversaries $\mathcal{A}$, the following advantage is negligible,

$$\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) := \left|\mathsf{Pr}\left[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_s, [\mathbf{Aw}]_s) = 1\right] - \mathsf{Pr}\left[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_s, [\mathbf{u}]_s) = 1\right]\right|$$

$\mathcal{D}_k$-KerMDHAssumption. Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1,2\}$. We say that the $\mathcal{D}_k$-*Kernel Diffie-Hellman assumption* holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ and all p.p.t adversaries $\mathcal{A}$, the following advantage is negligible,

$$\mathbf{Adv}^{\mathsf{KerMDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) := \mathsf{Pr}\left[[\mathbf{c}]_{3-s} \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, [\mathbf{A}]_s) : \mathbf{c}^\intercal \mathbf{A} = 0 \wedge \mathbf{c} \neq 0\right]$$

Lemma 2 ($\mathcal{D}_k$-MDDH $\Rightarrow$ $\mathcal{D}_k$-KerMDH) Let $k \in \mathbb{N}$ and let $\mathcal{D}_k$ be a matrix distribution. For any p.p.t adversary $\mathcal{A}$, there exists a p.p.t adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathsf{KerMDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) \le \mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{B})$.

As stated in [GHKP18], for $Q \in \mathbf{N}$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}$ and $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times Q}$, we consider the Q-fold $\mathcal{D}_k$-MDDH assumption, which states that distinguishing tuples of the form $([\mathbf{A}]_s, [\mathbf{AW}]_s)$ from $([\mathbf{A}]_i, [\mathbf{U}]_i)$ is hard. That is, a challenge for the Q-fold $\mathcal{D}_k$-MDDH assumption consists of Q independent challenges of the $\mathcal{D}_k$-MDDH assumption (with the same $\mathbf{A}$ but different randomness $\mathbf{w}$). In [EHK+13] it is shown that the two problems are equivalent, where the reduction loses at most a factor $(k+1)-k$.

Lemma 3 (Random self-reducibility of $\mathcal{D}_k$-MDDH [EHK+13]) Let $k, Q \in \mathbb{N}$ with $Q > 1$ and $s \in \{1,2,T\}$. For any p.p.t adversary $\mathcal{A}$, there exists and adversary $\mathcal{B}$ s.t. $T(\mathcal{B}) \approx T(\mathcal{A}) + Q \cdot poly(\lambda)$ with $poly(\lambda)$ independent of $T(\mathcal{A})$, and

$$\mathbf{Adv}^{Q-\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{A}}(\lambda) \le \mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{B}}(\lambda) + \tfrac{1}{p-1}.$$

Here

$$\mathbf{Adv}^{Q-\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{A}}(\lambda) := \left|\mathsf{Pr}\left[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_s, [\mathbf{AW}]_s) = 1\right] - \mathsf{Pr}\left[\mathcal{A}(\mathsf{BG}, [\mathbf{A}]_s, [\mathbf{U}]_s) = 1\right]\right.$$

where the probability is taken over $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{W} \leftarrow \mathbb{Z}_p^{k\times Q}$, $\mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times Q}$.

Pedersen commitment. The Pedersen commitment [Ped92] is a non-interactive commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, which consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda)$. On input the security parameter λ, generates a group $\mathbb{G}$ of prime order p with $\lceil \log_2 p \rceil = \lambda$, picks a generator g and $td \xleftarrow{\$} \mathbb{Z}_p^*$. It outputs public parameters $\mathsf{pp} = (\mathbb{G}, p, g, h = g^{td})$ and trapdoor information td.
- $\mathsf{Commit}(\mathsf{pp}, m)$. On input the public parameters pp and a message $m \in \mathbb{Z}_p$ outputs a commitment-opening pair $(c, d) \leftarrow (g^m h^r, r)$.
- $\mathsf{Verify}(\mathsf{pp}, c, d, m)$. On input $d = r$ outputs 1 if and only if $g^m h^r = c$.

The Pedersen commitment scheme is correct, perfectly hiding and computationally binding under the DL assumption in $\mathbb{G}$.

B Security proofs of our SCDS scheme

Theorem 1. The SCDS construction from Figure 2 is correct and perfectly hiding. Furthermore, if the q-co-DL (resp. q-co-GSDH) assumption holds, SCDS is computationally binding (resp. sound).

Proof. Correctness: We refer the reader to [FHS19](Th. 3) which already proved almost verbatim the same claim.

Hiding: We refer the reader to [FHS19](Th. 4) which already proved almost verbatim the same claim.

Binding: Following the approach from [FHS19], we consider the view of an unbounded adversary $\mathcal{A}$ in the hiding experiment and assume w.l.o.g that every query $\mathcal{S}$ to the $\mathcal{O}_{\mathsf{SS}}$ oracle satisfies $\mathcal{S} \subset \mathbb{Z}_p$ and $\emptyset \neq \mathcal{S} \subseteq (\mathcal{X}_0 \cap \mathcal{X}_1)$. Similarly, every query $\mathcal{D}$ to the $\mathcal{O}_{\mathsf{DS}}$ oracle satisfies $\mathcal{D} \subset \mathbb{Z}_p$ and $\emptyset \neq \mathcal{D} \cap \{\mathcal{X}_0 \cup \mathcal{X}_1\} = \emptyset$. To prove perfect hiding, the results from the adversary queries should be independent from b. In the following, we will prove that this is the case for the queries made to the oracle $\mathcal{O}_{\mathsf{DS}}$. We omit to prove here the case for queries made to $\mathcal{O}_{\mathsf{SS}}$ as the corresponding proof can be found almost verbatim in [FHS19](Th. 6).
(1) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ with $s \in \mathcal{X}_0 \cap \mathcal{X}_1$. Note that for all queries $\mathcal{D}_j$, we have $s \notin \mathcal{D}_j$ and for both $b \in \{0,1\}$, $C_b = r_b P_1$ is uniformly random in $\mathbb{G}_1^*$ for some $r_b \in Z_p^*$. Furthermore, jth query $\mathcal{D}_j$ to $\mathcal{O}_{\mathsf{DS}}$ is answered with $\underline{\mathsf{wit}}_{j,b} = (\gamma P_2, \frac{1-\gamma \cdot r_b}{\mathsf{Ch}_{\mathcal{D}_j}(s)} P_1, \pi_Q, Q)$ for a uniformly random $\gamma \in \mathbb{Z}_p^*$, so it does not depend on the bit b. Thus it is information-theoretically hidden.
(2) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ s.t. s is contained in one of the sets; say $s \in \mathcal{X}_0$. As in the previous case, for all queries $\mathcal{D}_j$, we have $s \notin \mathcal{D}_j$. If $b = 0$ then $\mathcal{A}$ receives a uniformly random $C_0 = r_0 P_1$ in $\mathbb{G}_1^*$ for some $r_0 \in Z_p^*$, and when it queries $\mathcal{D}_j$ to the $\mathcal{O}_{\mathsf{DS}}$ oracle, it receives $\underline{\mathsf{wit}}_{j,0} = (\gamma P_2, \frac{1-\gamma \cdot r_0}{\mathsf{Ch}_{\mathcal{D}_j}(s)} P_1, \pi_Q, Q)$ for a uniformly random $\gamma \in \mathbb{Z}_p^*$. If $b = 1$ then $\mathcal{A}$ receives $C_1 = \mathsf{Ch}_{\mathcal{X}_1}(s) \cdot r_1 P_1$ for a random $r_1 \in Z_p^*$ and the jth query $\mathcal{D}_j$ to $\mathcal{O}_{\mathsf{DS}}$ is answered with $\underline{\mathsf{wit}}_{j,1} = (q_1'(s).\frac{1}{r_1} P_2, q_2'(s) P_1, \pi_Q, Q)$. In this case, $q_1'(s) = q_1(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{X}_1}(s)$, and $q_2'(s) = q_2(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{D}_j}(s)$ for a random $\gamma \in \mathbb{Z}_p^*$ so both witnessess $\underline{\mathsf{wit}}_{j,0}$ and $\underline{\mathsf{wit}}_{j,1}$ are indistinguishable and do not depend on the bit b. Therefore, b is information-theoretically hidden from $\mathcal{A}$.
(3) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ with $s \notin \mathcal{X}_0 \cup \mathcal{X}_1$. For both $b \in \{0,1\}$: $C_b = P_1^{\mathsf{Ch}_{\mathcal{X}_b}(s) \cdot r_b}$ for a random $r_b \in Z_p^*$. If $s \notin \mathcal{D}_j'$ the jth query is answered with $\underline{\mathsf{wit}}_{j,b} = (q_1'(s).\frac{1}{r_b} P_2, q_2'(s) P_1, \pi_Q, Q)$. If $s \in \mathcal{D}_j'$: the jth query is answered with $\underline{\mathsf{wit}}_{j,b} = (q_1'(s).\frac{1}{r_b} P_2, q_2'(s) P_1, \pi_Q, Q)$. Observe that in both cases the first witness component $q_1'(s).\frac{1}{r_b} P_2$ perfectly hides b because $q_1'(s) = q_1(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{X}_b}(s)$ is uniformly random for $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$. Similarly, the second component $q_2'(s) = q_2(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{D}_j}(s)$ is also uniformly random. We conclude that b is information theoretically hidden from $\mathcal{A}$.

Soundness: We prove both equations in the soundness definition by reduction to the q-co-GSDH assumption. To do so, we consider an adversary $\mathcal{B}$ which on input an instance $I = (\mathsf{BG}, s)$, sets $\mathsf{pp} \leftarrow \mathsf{BG}$ and runs $\mathcal{A}(\mathsf{pp})$ in the soundness game.
(1) We assume that $\mathcal{A}$ is able to output $(C, O, \mathcal{X}, \mathcal{S}, \mathsf{wit})$ s.t. $\mathcal{X} \not\subseteq \mathcal{S}$, $\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1$ and $\mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wit}) = 1$.

Following the approach from [FHS19], we prove here the second inequality in the soundness definition considering an adversary $\mathcal{B}$ which on input an instance $I = (\mathsf{BG}, s)$, sets $\mathsf{pp} \leftarrow \mathsf{BG}$ and runs

$\mathcal{A}(\mathsf{pp})$ in the soundness game. We omit to prove here the first inequality as the corresponding proof can be found almost verbatim in [FHS19](Th. 5).
(2) We now assume $\mathcal{A}$ is able to output $(C, O, \mathcal{X}, \mathcal{D}, \underline{\mathsf{wit}})$ s.t. $\mathcal{X} \cap \mathcal{D} \neq \emptyset$, $\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1$ and $\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \underline{\mathsf{wit}}) = 1$.
(2.1) $s \notin \{\mathcal{X} \cup \mathcal{D}\}$: In this case, observe that $\exists\ c \in \mathbb{Z}_p^*$ s.t $s \neq c \in \mathcal{X} \cap \mathcal{D}$. Hence, the adversary can compute polynomials $q_1(s)$ and $q_2(s)$ s.t $\mathsf{Ch}_{\mathcal{X}}(s) = (c+X)q_1(s)$ and $\mathsf{Ch}_{\mathcal{D}}(s) = (c+X)q_2(s)$. Since $\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \underline{\mathsf{wit}}) = 1$, we have that:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(\mathsf{Ch}_{\mathcal{X}}(s) \cdot rP_1, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e((c+s)q_1(s) \cdot rP_1, w_0) \cdot e(w_1, (c+s)q_2(s)P_2) \\ &= e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2)^{(c+s)} \end{aligned}$$

Hence we have $e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2) = e(P_1, P_2)^{\frac{1}{(c+s)}}$. $\mathcal{A}$ is able to efficiently compute $q_1(s)$ and $q_2(s)$, and so the left side of the last equation can also be efficiently computed by $\mathcal{A}$. It follows that $\mathcal{B}$ can output the pair $(c, e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2))$ to break the q-co-GSDH assumption.
(2.2) $s \in \{\mathcal{X} \cap \mathcal{D}\}$: We have that $C = \gamma P_1$ for a random $\gamma \in \mathbb{Z}_p^*$ and that s is a root of $\mathsf{Ch}_{\mathcal{D}}(s)$. Therefore, the verification equation can be written as follows:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(C, w_0) \cdot e(w_1, [\mathsf{Id}]_2) \\ &= e(C \cdot w_1, w_0) \end{aligned}$$

Since $\mathcal{B}$ can efficiently compute the right side of the previous equation, $\mathcal{A}$ can also output a solution $(c, e(C \cdot w_1, w_0)^{\frac{1}{c+s}})$ to the q-co-GSDH problem.
(2.3) $s \in \mathcal{X} \ \wedge \ s \notin \mathcal{D}$: As before, $C = \gamma P_1$ for a random $\gamma \in \mathbb{Z}_p^*$, but we also have that $\exists\ c \in \mathbb{Z}_p^*$ s.t $s \neq c \in \mathcal{X} \cap \mathcal{D}$, and we can write $\mathsf{Ch}_{\mathcal{D}}(s)=(c+X)q_1(s)$. The verification equation can then be re-stated as:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(\gamma P_1, w_0) \cdot e(w_1, (c+s)q_1(s)P_2) \\ &= e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2)^{(c+s)} \end{aligned}$$

Hence, we have $e(P_1, P_2)^{\frac{1}{(c+s)}} = e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2)$, where the right side can be efficiently computed by $\mathcal{A}$. Therefore, $\mathcal{B}$ can output a solution $(c, e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2))$ to the q-co-GSDH problem.
(2.4) $s \notin \mathcal{X} \ \wedge \ s \in \mathcal{D}$: In this case we have that $C = \mathsf{Ch}_{\mathcal{D}}(s) \cdot rP_1$ with $\mathsf{Ch}_{\mathcal{D}}(s) = (c+X)q_1(s)$, for some $s \neq c \in \{\mathcal{X} \cap \mathcal{D}\}$. Again, c and q_1 can be efficiently computed and the verification equation can then re-stated as:

$$\begin{aligned} e(P_1, P_2) &= e(\mathsf{Ch}_{\mathcal{X}}(s) \cdot rP_1, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e((c+s)q_1(s) \cdot rP_1, w_0) \cdot e(w_1, P_2^0) \\ &= e(q_1(s) \cdot rP_1 \cdot w_1, w_0)^{(c+s)} \end{aligned}$$

Hence, we have $e(P_1, P_2)^{\frac{1}{(c+s)}} = e(q_1(s) \cdot rP_1 \cdot w_1, w_0)$, where the right side can be efficiently computed by $\mathcal{A}$. Therefore, $\mathcal{B}$ can output a solution $(c,\ e(q_1(s) \cdot rP_1 \cdot w_1,\ w_0))$ to the q-co-GSDH problem. □

C Security proofs of our Malleable NIZK Argument

Theorem 2. The protocol in Figure 3 is a fully adaptive NIZK argument for the OR-language $\mathcal{L}^{\vee}_{\mathbf{A}_0, \mathbf{A}_1}$ if the falsifiable $\mathcal{L}_1$-$(4k+1)$-extKerMDH assumption holds in $\mathbb{G}_2$.

Proof. We have to show *completeness, perfect zero-knowledge, computational soundness* and *derivation privacy*. We do it in the same way as done for the original protocols from [CH20] (Theorem 19) and [KSD19] (Theorem 1).

Perfect Completeness: Let $[\mathbf{x}_1]_1 = [\mathbf{A}_0]_1\mathbf{w}_1$ and $[\mathbf{x}_2]_1 = [\mathbf{A}_0]_1\mathbf{w}_2$ be valid statements for $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ with witnesses $\mathbf{w}_1$ and $\mathbf{w}_2$, and let $\pi^1=(([\mathbf{a}_i]^1_1, [\mathbf{d}_i]^1_2, [z_i]_2)_{i\in\{0,1\}}, P)$ and $\pi^2 = (([\mathbf{a}_i]^2_1, [\mathbf{d}_i]^2_2, [z_i]_2)_{i\in\{0,1\}}, P)$ be valid proofs for $[\mathbf{x}_1]_1, [\mathbf{x}_2]_1 \in \mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$. Let $\widehat{\pi} = (\alpha[\mathbf{a}_0]^1_1+\alpha\beta[\mathbf{a}_0]^2_1, \alpha[\mathbf{a}_1]^1_1+\alpha\beta[\mathbf{a}_1]^2_1, \alpha[\mathbf{d}_0]^1_2+\alpha\beta[\mathbf{d}_0]^2_2, \alpha[\mathbf{d}_1]^1_2 + \alpha\beta[\mathbf{d}_1]^2_2, \alpha[z_0]_2, \alpha[z_1]_2, \alpha P)$ be the output from $\mathsf{PPrv}(crs, [\mathbf{x}_1]_1, \mathbf{w}_1, [\mathbf{x}_2]_1, \mathbf{w}_2)$, where the corresponding witness is $\widehat{\mathbf{w}} = \mathbf{w}_1 + \beta\mathbf{w}_2$. In the following we show that $\mathsf{PRVer}(crs, \widehat{[\mathbf{x}]_1} = [\mathbf{x}_1]_1 + \beta[\mathbf{x}_2]_1, \widehat{\pi}) = 1$.

First, we have that

$$
\begin{aligned}
& \mathbf{A}_0 \cdot \widehat{d_0} = \widehat{x} \cdot \widehat{z_0} + \widehat{\mathbf{a}_0} \\
\iff\ & \mathbf{A}_0(\alpha[\mathbf{d}_0]^1_2 + \alpha\beta[\mathbf{d}_0]^2_2) = \widehat{x} \cdot \widehat{z_0} + (\alpha[\mathbf{a}_0]^1_1 + \alpha\beta[\mathbf{a}_0]^2_1) \\
\overset{[\mathbf{a}_0]^i_1=\mathbf{A}_0\cdot\mathbf{s}_i}{\iff}\ & \mathbf{A}_0(\alpha[\mathbf{d}_0]^1_2 + \alpha\beta[\mathbf{d}_0]^2_2) = \widehat{x} \cdot \widehat{z_0} + (\alpha\mathbf{A}_0 \cdot \mathbf{s}_1 + \alpha\beta\mathbf{A}_0 \cdot \mathbf{s}_2) \\
\overset{[\mathbf{d}_0]^i_1=z_0\mathbf{w}_i+\mathbf{s}_i}{\iff}\ & \mathbf{A}_0(\alpha(z_0\mathbf{w}_1 + \mathbf{s}_1) + \alpha\beta(z_0\mathbf{w}_2 + \mathbf{s}_2)) = \widehat{x} \cdot \widehat{z_0} + \mathbf{A}_0(\alpha\mathbf{s}_1 + \alpha\beta\mathbf{s}_2) \\
\iff\ & \mathbf{A}_0(\alpha z_0\mathbf{w}_1 + \alpha\beta z_0\mathbf{w}_2 + (\alpha\mathbf{s}_1 + \alpha\beta\mathbf{s}_2)) = \widehat{x} \cdot \widehat{z_0} + \mathbf{A}_0(\alpha\mathbf{s}_1 + \alpha\beta\mathbf{s}_2) \\
\iff\ & \mathbf{A}_0(\alpha z_0\mathbf{w}_1 + \alpha\beta z_0\mathbf{w}_2) + \mathbf{A}_0(\alpha\mathbf{s}_1 + \alpha\beta\mathbf{s}_2) = \widehat{x} \cdot \widehat{z_0} + \mathbf{A}_0(\alpha\mathbf{s}_1 + \alpha\beta\mathbf{s}_2) \\
\overset{\widehat{x}=\mathbf{A}_0\widehat{w}}{\iff}\ & \mathbf{A}_0(\alpha z_0\mathbf{w}_1 + \alpha\beta z_0\mathbf{w}_2) = \mathbf{A}_0 \cdot \widehat{w} \cdot \widehat{z_0} \\
\overset{\widehat{z_0}=\alpha z_0}{\iff}\ & \alpha z_0(\mathbf{w}_1 + \beta\mathbf{w}_2) = \alpha z_0\widehat{w} \\
\iff\ & \mathbf{w}_1 + \beta\mathbf{w}_2 = \widehat{w}
\end{aligned}
$$

Similarly, we have that

$$
\begin{aligned}
& \mathbf{A}_1 \cdot \widehat{d_0} = \widehat{x} \cdot \widehat{z_1} + \widehat{\mathbf{a}_1} \\
\iff\ & \mathbf{A}_1(\alpha[\mathbf{d}_1]^1_2 + \alpha\beta[\mathbf{d}_1]^2_2) = \widehat{x} \cdot \widehat{z_1} + (\alpha[\mathbf{a}_1]^1_1 + \alpha\beta[\mathbf{a}_1]^2_1) \\
\overset{[\mathbf{a}_1]^i_1=\mathbf{A}_1\mathbf{d}^i_1-z_1\mathbf{x}_i}{\iff}\ & \mathbf{A}_1(\alpha[\mathbf{d}_1]^1_2 + \alpha\beta[\mathbf{d}_1]^2_2) = \widehat{x} \cdot \widehat{z_1} + (\alpha(\mathbf{A}_1\mathbf{d}^1_1 - z_1\mathbf{x}_1) \\
& + \alpha\beta(\mathbf{A}_1\mathbf{d}^2_1 - z_1\mathbf{x}_2)) \\
\iff\ & \alpha\mathbf{A}_1[\mathbf{d}_1]^1_2 + \alpha\beta\mathbf{A}_1[\mathbf{d}_1]^2_2 = \widehat{x} \cdot \widehat{z_1} + \alpha(\mathbf{A}_1\mathbf{d}^1_1) - \alpha z_1\mathbf{x}_1 \\
& + \alpha\beta(\mathbf{A}_1\mathbf{d}^2_1) - \alpha\beta z_1\mathbf{x}_2 \\
\overset{\widehat{z_1}=\alpha z_1}{\iff}\ & \alpha\mathbf{A}_1[\mathbf{d}_1]^1_2 + \alpha\beta\mathbf{A}_1[\mathbf{d}_1]^2_2 = \widehat{x} \cdot \alpha z_1 - (\mathbf{x}_1 + \beta\mathbf{x}_2)\alpha z_1 \\
& + \alpha\mathbf{A}_1[\mathbf{d}_1]^1_2 + \alpha\beta\mathbf{A}_1[\mathbf{d}_1]^2_2 \\
\overset{\widehat{x}=\mathbf{x}_1+\beta\mathbf{x}_2}{\iff}\ & \alpha\mathbf{A}_1[\mathbf{d}_1]^1_2 + \alpha\beta\mathbf{A}_1[\mathbf{d}_1]^2_2 = \alpha\mathbf{A}_1[\mathbf{d}_1]^1_2 + \alpha\beta\mathbf{A}_1[\mathbf{d}_1]^2_2
\end{aligned}
$$

Finally, we also have that $e(P, [z]_2) = [\alpha z]_T = e([1]_1, [\alpha z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$.

Perfect Zero-Knowledge: We have to show that the distributions PSim and PPrv are identical. As in [CH20] (Theorem 19), this follows from the fact that the simulator is able to generate an identically distributed proof when given the trapdoor e, while also hiding the value α used to randomize the challenges with P as done by the real prover.

Computational Soundness: Again, the only difference from [CH20] (Theorem 19) is that we now use a linear relation to check the challenge, which is equivalent to verifiy the original equation $z = z_0 + z_1$ and therefore soundness can be proven following the proof from [CH20] almost in verbatim.

Derivation Privacy: As in [KSD19], the algorithm ZKEval outputs a proof with new independent randomness, which has an idientical distribution compared to the PProv when computing a single proof. Thus, we also achieve perfect derivation privacy. □

$\underline{\mathsf{Exp}_0^{\mathsf{core}}(\lambda), \beta \in \{0,1\}}$:	$\underline{\mathsf{TAGO}()}$:
$\mathsf{ctr} \leftarrow 0$	$\mathsf{ctr} \leftarrow \mathsf{ctr}+1;\ r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$	$[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1,\ [\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
$\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$	$\Omega = (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1) \leftarrow \mathsf{PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
$(\mathsf{crs}, \mathsf{td}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$	$[\mathbf{u}']_1 \leftarrow (\mathbf{k}_0 + \beta \cdot \mathbf{RF}(\mathsf{ctr}))^\top [\mathbf{t}]_1;\ [\mathbf{u}'']_1 \leftarrow (\mathbf{k}_0 + \beta \cdot \mathbf{k}_1)^\top [\mathbf{w}]_1$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$	$\mathsf{tag} \leftarrow ([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$
$\mathbf{k}_0, \mathbf{k}_1 \xleftarrow{\$} \mathbb{Z}_p^2$	**return** tag
$\mathsf{tag} \leftarrow \mathcal{A}^{\mathsf{TAGO}()}(\mathsf{pp})$	
return $\mathsf{VERO}(\mathsf{tag})$	$\underline{\mathsf{VERO}(\mathsf{tag})}$:
	parse $\mathsf{tag} = ([\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1, [\mathbf{u}']_1)$
	if $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, Z_1)) \wedge$
	$\exists\ \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}']_1 = (\mathbf{k}_0 + \beta \cdot \mathbf{RF}(\mathsf{ctr}'))^\top [\mathbf{t}]_1$
	return 1 **else return** 0

Fig. 7: Core lemma for our SPS-EQ scheme.

D Security proofs of our SPS-EQ scheme

Lemma 4 (Core Lemma) If the $\mathcal{D}_1$-MDDH (DDH) assumption holds in $\mathbb{G}_1$ and the tuple of algorithms $\Pi = (\mathsf{PGen}, \mathsf{PPro}, \mathsf{PSim}, \mathsf{PRVer})$ is a non-interactive zero-knowledge argument for $\mathcal{L}^{\vee}_{\mathbf{A}_0, \mathbf{A}_1}$, then going from experiment $\mathsf{Exp}_0^{\mathsf{core}}$ to $\mathsf{Exp}_1^{\mathsf{core}}$ (Fig. 7) can (up to negligible terms) only increase the winning chance of an adversary. More precisely, for every adversary $\mathcal{A}$, there exists adversaries $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{B}_2$ s.t

$$\mathbf{Adv}_0^{\mathsf{core}}(\mathcal{A}) - \mathbf{Adv}_1^{\mathsf{core}}(\mathcal{A}) \leq \Delta_{\mathcal{A}}^{\mathsf{core}}, \text{ where}$$

$$\Delta_{\mathcal{A}}^{\mathsf{core}} = (2 + 2\lceil \log Q \rceil \mathbf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{B}) + (8\lceil \log Q \rceil + 4)\mathbf{Adv}_{\mathcal{D}_1, \mathbb{G}_s}^{\mathsf{MDDH}}(\mathcal{B}_1)$$
$$2\lceil \log Q \rceil \mathbf{Adv}_{\Pi}^{\mathsf{snd}}(\mathcal{B}_2) + \lceil \log Q \rceil \Delta_{\mathcal{D}_1} + \frac{(8\lceil \log Q \rceil + 4)}{p-1} + \frac{(\lceil \log Q \rceil)Q}{p}$$

and the term $\Delta_{\mathcal{D}_1}$ is statistically small.

Proof. The proof of this lemma is very similar (in parts verbatim) to the one given in [KSD19], which in turns extends the original core lemma from [GHKP18]. The main difference is that we use the standard definition for zero-knowledge in our NIZK argument system instead of the composable one. As pointed out in [AJO+19], the standard notion of zero-knowledge suffices in this context. For completeness, we present the full proof below.

Game 0: We define **Game 0**$= \mathsf{Exp}_0^{\mathsf{core}}$ ad thus by definition:

$$\mathbf{Adv}_0 = \mathbf{Adv}_0^{\mathsf{core}}(\mathcal{A})$$

Game 1: In this game we replace PPro with PSim in **Game 0** to compute the proof. An adversary $\mathcal{B}$ for **Game 1** is such that $T(\mathcal{B}) \approx T(\mathbf{A}) + Q \cdot poly(\lambda)$ and

$$\mathbf{Adv}_0 - \mathbf{Adv}_1 \leq \mathbf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{B})$$

where $\mathbf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{B})$ is the advantage of $\mathcal{B}$ ito break the zero-knowledge property from Π.

Game 2: In this game we pick $[\mathbf{t}]_1, [\mathbf{w}]_1 \xleftarrow{\$} \mathbb{G}_1^2$ instead of computing them as in the previous game. We can switch $[\mathbf{t}]_1$ and $[\mathbf{w}]_1$ to random over $\mathbb{G}_1^2$ by applying the $\mathcal{D}_1$-MDDH assumption. More precisely, let $\mathcal{A}$ be an adversary distinguishing between **Game 1** and **Game 2** and let $\mathcal{B}_1$ be an adversary given two Q-fold $\mathcal{D}_1$-MDDH challenges $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}_1]_1, ..., [\mathbf{q}_Q]_1)$ and $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}'_1]_1, ..., [\mathbf{q}'_Q]_1)$ as input. Now $\mathcal{B}_1$ sets up the game for $\mathcal{A}$ similar to **Game 1**, but instead choosing $\mathbf{A}_0 \xleftarrow{\$} \mathcal{D}_1$, it uses its challenge matrix $[\mathbf{A}_0]_1$ as part of the public parameters pp. Further, to answer tag queries $\mathcal{B}_1$ sets $[\mathbf{t}_i]_1 := [\mathbf{q}_i]_1$, and $[\mathbf{w}_i]_1 := [\mathbf{q}_i]_1$ and computes the rest accordingly. This is possible as the proof Ω is simulated from **Game 1** on. In case $\mathcal{B}_1$ was given a real $\mathcal{D}_1$-MDDH challenge, it simulates **Game 1** and otherwise **Game 2**. Thus, by lemma 3, we have an adversary $\mathcal{B}_1$ with $T(\mathcal{B}_1) \approx T(\mathcal{A}) + Q \cdot poly(\lambda)$ and

$$\mathbf{Adv}_1 - \mathbf{Adv}_2 \leq 2\mathbf{Adv}_{\mathcal{D}_1, \mathbb{G}_s}^{\mathsf{MDDH}}(\mathcal{B}_1) + \tfrac{2}{p-1}$$

Game 3.0: Let us denote a sequence of games with $3.i$, where $\mathbf{RF}_i$ is the random function $\mathbf{RF}$ on i-bit prefixes, and the i-bit prefix of ctr is $\mathsf{ctr}_{|i}$. In this game, we compute $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbf{RF}_i(\mathsf{ctr}_{|i})[\mathbf{t}]_1$, and $[\mathbf{u}'']_1 = (\mathbf{k}_0 + \mathbf{k}'_0)[\mathbf{w}]_1$ (where $\mathbf{k}'_0 = \mathbf{RF}_0(\mathsf{ctr}_{|0})$). In the verification algorithm also, we verify $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbf{RF}_i(\mathsf{ctr}|i')[\mathbf{t}]_1$ for $\mathsf{ctr}' \leq \mathsf{ctr}$, and $[\mathbf{u}'']_1 = (\mathbf{k}_0 + \mathbf{k}'_0)[\mathbf{w}]_1$. As for all $\mathsf{ctr} \in \mathbb{N}$ we have $\mathbf{RF}_0(\mathsf{ctr}_{|0}) = \mathbf{RF}_0(\epsilon)$ and $\mathbf{k}_0$ is identically distributed to $\mathbf{k}_0 + \mathbf{RF}_0(\epsilon)$ for $\mathbf{k}_0 \xleftarrow{\$} \mathbb{Z}_p$, we have

$$\mathbf{Adv}_{3.0} = \mathbf{Adv}_2$$

Game 3.i→Game 3.(i+1) We proceed via a series of hybrid games $H_{i.j}$ for $i \in [0,\log(Q)-1]$ and $j \in [1,8]$, marking the adversary's advantage on each game with $\mathbf{Adv}'$.
Game 3.i→$H_{i.1}$: In this game, we compute $[\mathbf{t}]_1 = [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_{1.i}$ and $[\mathbf{w}]_1 = [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_{2.i}$, instead of picking them randomly. Here, ctr_{i+1} is the $i+1$'st bit of the binary representation of ctr. More precisely, we introduce an intermediary game $H_{i.0}$, where we choose $[\mathbf{t}_i]_1$ and $[\mathbf{w}_i]_1$ as

$$[\mathbf{t}_i]_1 = \begin{cases} [\mathbf{A}_{\mathsf{ctr}_{i+1}}] r_{1.i} \text{ for } r_{1.i} \xleftarrow{\$} \mathbb{Z}_p & \text{if } \mathsf{ctr}_{i+1} = 0 \\ [\mathbf{u}_i]_1 \text{ for } \mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^2 & \text{otherwise} \end{cases}$$

$$[\mathbf{w}_i]_1 = \begin{cases} [\mathbf{A}_{\mathsf{ctr}_{i+1}}] r_{2.i} \text{ for } r_{2.i} \xleftarrow{\$} \mathbb{Z}_p & \text{if } \mathsf{ctr}_{i+1} = 0 \\ [\mathbf{u}'_i]_1 \text{ for } \mathbf{u}'_i \xleftarrow{\$} \mathbb{Z}_p^2 & \text{otherwise} \end{cases}$$

Let $\mathcal{A}$ be an adversary distinguishing between **Game 3.i** and $H_{i.0}$ and let $\mathcal{B}_1$ be an adversary given two Q-fold $\mathcal{D}_1$-MDDH challenges $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}_1]_1, ..., [\mathbf{q}_Q]_1)$ and $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}'_1]_1, ..., [\mathbf{q}'_Q]_1)$. Then $\mathcal{B}_1$ sets up the game for $\mathcal{A}$ similar to **Game 3.i**, where it embeds $[\mathbf{A}_0]_1$ into the public parameters pp. Further, whenever obtaining a simulation query ctr with $\mathsf{ctr}_{i+1} = 0$, $\mathcal{B}_1$ sets $[\mathbf{t}_i]_1 := [\mathbf{q}_i]_1$ and $[\mathbf{w}_i]_1 := [\mathbf{q}'_i]_1$ and otherwise follows **Game 3.i**. Similarly, we can reduce the transition from game $H_{i.0}$ to $H_{i.1}$ to the MDDH assumption. We have

$$|\mathbf{Adv}_{3.i} - \mathbf{Adv}'_{i.1}| \leq 4\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_1) + \tfrac{4}{p-1}$$

$H_{i.1}$→$H_{i.2}$: In this step we reverse the transition from **Game 0** to **Game 1** and thus replace PSim with PPro from game $H_{i.1}$ on. We choose all $[\mathbf{t}]_1$, $[\mathbf{w}]_1$ in tag queries from $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ with corresponding witness and can thus honestly generate proofs. Therefore,

$$|\mathbf{Adv}'_{i.2} - \mathbf{Adv}'_{i.1}| \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.2}$→$H_{i.3}$: From game $H_{i.3}$ on we introduce an additionally check in the verification oracle. Namely, VERO checks that $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \text{span}([\mathbf{A}_0]_1) \vee \text{span}([\mathbf{A}_1]_1)$. We can employ the soundness of Π to obtain

$$|\mathbf{Adv}'_{i.3} - \mathbf{Adv}'_{i.2}| \leq \mathbf{Adv}^{\mathsf{snd}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.3}$→$H_{i.4}$: Let $\mathbf{A}_0^{\perp} \in \text{orth}(\mathbf{A}_0)$ and $\mathbf{A}_1^{\perp} \in \text{orth}(\mathbf{A}_1)$. We introduce an intermediary game $H_{i.3.1}$, where we replace the random function $\mathbf{RF}_i : \{0,1\}^i \to \mathbb{Z}_p^2$ by

$$\mathbf{RF}'_i : \{0,1\}^i \to \mathbb{Z}_p^2,\ \mathbf{RF}'_i(v) := (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma_i(v)\ \Upsilon_i(v))^{\top}$$

where $v \leftarrow \{0,1\}^i$ is an i-bit string and $\Gamma_i, \Upsilon_i : \{0,1\}^i \to \mathbb{Z}_p$ are two independent random functions. With probability $1 - \Delta_{\mathcal{D}_1}$ the matrix $(\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})$ has full rank. In this case, going from game $H_{i.3}$ to game $H_{i.3.1}$ consists merely in a change of basis, thus, these two games are perfectly indistinguishable. We obtain

$$|\mathbf{Adv}'_{i.3.1} - \mathbf{Adv}'_{i.3}| \leq \Delta_{\mathcal{D}_1}$$

We now define $\mathbf{RF}_{i+1} : \{0,1\}^{i+1} \to \mathbb{Z}_p^2$ s.t,

$$\mathbf{RF}_{i+1}(v) := \begin{cases} (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma'_i(v_{|i})\ \Upsilon_i(v_{|i}))^{\top} & \text{if } v_{|i+1} = 0 \\ (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma_i(v_{|i})\ \Upsilon'_i(v_{|i}))^{\top} & \text{otherwise} \end{cases}$$

where $\Gamma'_i, \Upsilon'_i : \{0,1\}^i \to \mathbb{Z}_p$ are fresh independent random functions. Now $\mathbf{RF}_{i+1}$ constitutes a random function $\{0,1\}^{i+1} \to \mathbb{Z}_p^2$. Replacing $\mathbf{RF}'_{i+1}(\mathsf{ctr}_{|i})$ by $\mathbf{RF}_{i+1}(\mathsf{ctr}_{|i+1})$ does not show up in any of the tag queries, as we have

$$\mathbf{RF}_{i+1}(\mathsf{ctr}_{|i+1})^{\top}[\mathbf{t}]_1 = \mathbf{RF}_{i+1}(\mathsf{ctr}_{|i+1})^{\top}[\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_1 = ... = \mathbf{RF}'_i(\mathsf{ctr}_{|i})^{\top}[\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_1$$

In the verification oracle we check $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \text{span}([\mathbf{A}_0]_1) \vee \text{span}([\mathbf{A}_1]_1)$. Let us define $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \text{span}(\mathbf{A}_0)$ and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \text{span}(\mathbf{A}_1)$, and replace $\mathbf{RF}_i(\mathsf{ctr}_{|i})$ by $\mathbf{RF}_{i+1}(\mathsf{ctr}_{|i}|d_{[\mathbf{t}]})$. Thus, by similar reasoning as for tag queries, the change does not show up in the final verification query either. We obtain

$$|\mathbf{Adv}'_{i.4} - \mathbf{Adv}'_{i.3}| \leq \Delta_{\mathcal{D}_1}$$

$H_{i.4} \rightarrow H_{i.5}$: From game $H_{i.5}$ on, we extend the set $\mathcal{S}$ in the verification oracle from $\mathcal{S}_{i.4} := \mathbf{RF}_{i+1}(\mathsf{ctr}'_{|i}|d_{[\mathbf{t}]}) : \mathsf{ctr}' \leq \mathsf{ctr}$ to $\mathcal{S}_{i.5} := \mathbf{RF}_{i+1}(\mathsf{ctr}'_{|i}|b) : \mathsf{ctr}' \leq \mathsf{ctr}, b \in \{0,1\}$. That is, we regard a verification query $([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$ as valid, if there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbf{RF}_{i+1}(\mathsf{ctr}'_{|i}|b)^\top [\mathbf{t}]_1$ for $b \in \{0,1\}$ arbitrary, instead of requiring $b = d_{[\mathbf{t}]}$. As changing the verification oracle does not change the view of the adversary before providing its output and as we have $\mathcal{S}_{i.4} \subseteq \mathcal{S}_{i.5}$, the transition from game $H_{i.4}$ to game $H_{i.5}$ can only increase the chance of the adversary. We thus have

$$\mathbf{Adv}'_{i.4} \leq \mathbf{Adv}'_{i.5}$$

$H_{i.5} \rightarrow H_{i.6}$: The difference between game $H_{i.5}$ and game $H_{i.6}$ is that in the latter we only regard a verification query $([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$ valid, if there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbf{RF}_{i+1}(\mathsf{ctr}'_{|i}|\mathsf{ctr}'_{|i+1})^\top [\mathbf{t}]_1$ (instead of allowing the last bit to be arbitrary). As the only way an adversary can learn the image of $\mathbf{RF}_{i+1}$ on a value is via tag queries and $\mathbf{RF}_{i+1}$ is a random function, a union bound over the elements in Q_{tag} yields

$$|\mathbf{Adv}'_{i.5} - \mathbf{Adv}'_{i.6}| \leq \frac{Q}{p}$$

$H_{i.6} \rightarrow H_{i.7}$: The oracle VERO does not perform the additional check $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \text{span}([\mathbf{A}_0]_1 \vee \text{span}([\mathbf{A}_1]_1)$ anymore from game $H_{i.7}$ on. This is justified by the soundness of Π. As in transition $H_{i.2} \rightarrow H_{i.3}$ we obtain

$$|\mathbf{Adv}'_{i.6} \leq \mathbf{Adv}'_{i.7}| \leq \mathbf{Adv}^{\mathsf{snd}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.7} \rightarrow H_{i.8}$: This transition is similar to transition **Game 0** to **Game 1**. For an adversary $\mathcal{B}_2$ we obtain

$$\mathbf{Adv}'_{i.7} - \mathbf{Adv}'_{i.8} \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.8} \rightarrow$ **Game 3.(i+1)**: We switch $[\mathbf{t}]_1, [\mathbf{w}]_1$ generated by TAGO to uniformly random over $\mathbb{G}_1^2$, using the MDDH assumption first on $[\mathbf{A}_0]_1$, then on $[\mathbf{A}_1]_1$. Similarly than for the transition **Game 3.i** $\rightarrow H_{i.1}$, we obtain

$$|\mathbf{Adv}_{3.(i+1)} - \mathbf{Adv}'_{i.8}| \leq 4\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_2) + \tfrac{4}{p-1}$$

Game 3.(log(Q)) $\rightarrow \mathsf{Exp}^{\mathsf{core}}_{1,\mathcal{A}}$: It is left to reverse the changes introduced in the transitions from **Game 0** to **Game 2** to end up at the experiment $\mathsf{Exp}^{\mathsf{core}}_{1,\mathcal{A}}$. In order to do so we introduce an intermediary **Game 4**, where we set $[\mathbf{t}]_1 := [\mathbf{A}_0]_1 r_1$ and $[\mathbf{w}]_1 := [\mathbf{A}_0]_1 r_2$ for $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. This corresponds to reversing transition **Game 1** to **Game 2**. By the same reasoning for every adversary $\mathcal{A}$ we thus obtain

$$|\mathbf{Adv}_{3.(\log Q)} - \mathbf{Adv}_4| \leq 2\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_1) + \tfrac{2}{p-1}$$

As $[\mathbf{t}]_1$, $[\mathbf{w}]_1$ are now chosen from $\text{span}([\mathbf{A}_0]_1)$ again, we have

$$\mathbf{Adv}_4 - \mathbf{Adv}^{\mathsf{core}}_1 \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

□

Theorem 4. If the KerMDH and MDDH assumptions hold, the SPS-EQ in Figure 4 is unforgeable.

Proof. We prove the claim by using a sequence of Games and we denote the advantage of the adversary in the j-th game as $\mathbf{Adv}_j$.

Game 0: This game is the original game and we have:

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{SPS-EQ}}(\mathcal{A})$$

Game 1: In this game, in Verify, we replace the verification in line (2:) with the following equation:

$$[\mathbf{u}^*_1]_1 = \mathbf{K}_0^\top [\mathbf{t}^*]_1 + \mathbf{K}^\top [\mathbf{m}^*]_1$$

For any signature $\sigma = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ that passes the original verification but not verification of **Game 1**, the value $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\mathbf{m}^*]_1$ is a non-zero vector in the kernel of $\mathbf{A}$. Thus, if $\mathcal{A}$ outputs such a signature, we can construct an adversary $\mathcal{B}$ that breaks the $\mathcal{D}_1$-KerMDH assumption in $\mathbb{G}_2$. First, the adversary $\mathcal{B}$ receives $(\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [z]_2)$, samples all other parameters and simulates **Game 1** for $\mathcal{A}$. When $\mathcal{B}$ receives the forgery from $\mathcal{A}$ as $\sigma = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ for $[\mathbf{m}^*]_1$, he passes the following values to its own challenger: $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\mathbf{m}^*]_1$. We have:

$$|\mathbf{Adv}_1 - \mathbf{Adv}_0| \leq \mathbf{Adv}^{\mathsf{KerMDH}}_{\mathcal{D}_1,\mathbb{G}_2}(\mathcal{A})$$

Game 2: In this game, we set $\mathbf{K}_0 = \mathbf{K}_0 + \mathbf{k}_0(\mathbf{a}^\perp)^\top$ (in the key generation we can pick $\mathbf{k}_0 \in \mathbb{Z}_p^2$ and $\mathbf{K}_0 \in \mathbb{Z}_p^{2\times 2}$ and set $\mathbf{K}_0$; we have $\mathbf{a}^\perp\mathbf{A} = 0$). We compute $[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\mathbf{m}]_1 + \mathbf{a}^\perp(\mathbf{k}_0)^\top[\mathbf{t}]_1$ and $[\mathbf{u}_2]_1 = \mathbf{K}_0^\top[\mathbf{w}]_1 + \mathbf{a}^\perp(\mathbf{k}_0)^\top[\mathbf{w}]_1$. There is no difference to the previous game since both are distributed identically. So, we have:

$$\mathbf{Adv}_2 = \mathbf{Adv}_1$$

Game 3: In this game, we add the part of $\mathbf{RF}(\mathsf{ctr})$ for $\mathsf{ctr} = \mathsf{ctr} + 1$, where $\mathbf{RF}$ is a random function, and obtain $[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\mathbf{m}]_1 + \mathbf{a}\perp(\mathbf{k}_0 + \mathbf{RF}(\mathsf{ctr}))^\top[\mathbf{t}] - 1$ and $[\mathbf{u}_2]_1 = \mathbf{K}\top_0[\mathbf{w}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{k}')^\top[\mathbf{w}]_1$. In the verification we have:

$$1 \leftarrow \mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, \pi)) \text{ and}$$
$$\exists\ \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{RF}(\mathsf{ctr}'))^\top + \mathbf{K}^\top[\mathbf{m}]_1$$

Let $\mathcal{A}$ be an adversary that distinguishes between **Game 3** and **Game 2**. We can construct an adversary $\mathcal{B}_1$ that breaks the core lemma. $\mathcal{B}_1$ receives $\mathsf{pp} = (BG, [\mathbf{A}_0]_1, \mathsf{crs})$ from $\mathsf{Exp}^{\mathsf{core}}_{\beta,\mathcal{B}_1}$. $\mathcal{B}_1$ picks $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$, $\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$, $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{2\times \ell}$, and sends public key $\mathsf{pk} = ([\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$ to $\mathcal{A}$. $\mathcal{B}_1$ uses the oracle $\mathsf{TAG}()$ to construct the signing algorithm. This oracle takes no input and returns $\mathsf{tag} = (([\mathbf{t}]_1, [\mathbf{w}]_1, (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1), [\mathbf{u}']_1, [\mathbf{u}'']_1)$. Then $\mathcal{B}_1$ computes $[\mathbf{u}_1]_1 = \mathbf{K}_0[\mathbf{t}]_1 + \mathbf{a}^\perp[\mathbf{u}']_1 + \mathbf{K}^\top[\mathbf{m}]_1$, $[\mathbf{u}_2]_1 = \mathbf{K}_0^\top[\mathbf{w}]_1 + \mathbf{a}^\perp[\mathbf{u}'']_1$, and sends the signature $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ and tag $\tau = ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$ to $\mathcal{A}$. When the adversary $\mathcal{A}$ sends the forgery $([\mathbf{m}^*]_1, \sigma^*) = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$, $\mathcal{B}$ returns 0 if $[\mathbf{u}_1]_1 = 0$; otherwise it checks whether there exists $[\mathbf{u}'^*]_1$ such that $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\mathbf{m}^*]_1 = \mathbf{a}^\perp[\mathbf{u}'^*]_1$. If it does not hold, then it returns 0 to $\mathcal{A}$, otherwise $\mathcal{B}_1$ computes $[\mathbf{u}'^*]_1$, and calls the verification oracle $\mathsf{VERO}()$ on the tag $\mathsf{tag}^* = ([\mathbf{u}'^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ and returns the answer to $\mathcal{A}$. Using the core lemma, we have:

$$\mathbf{Adv}_2 - \mathbf{Adv}_3 \leq \mathbf{Adv}^{\mathsf{core}}_{\mathsf{BG}}(\mathcal{B}_1)$$

Game 4: In this game, we pick r_1, r_2 from $\mathbb{Z}_p^*$ instead of $\mathbb{Z}_p$. The difference of advantage between **Game 3** and **Game 4** is bounded by the statistical distance between the two distributions of r_1, r_2. So, under Q adversarial queries, we have:

$$\mathbf{Adv}_4 - \mathbf{Adv}_3 \leq \frac{Q}{p}$$

Game 5: In this game, we pick $\tilde{\mathsf{ctr}} \xleftarrow{\$} [1, Q]$, and we add a condition $\mathsf{ctr}' = \tilde{\mathsf{ctr}}$ to verification. Actually, now we have this conditions:

$$1 \leftarrow \mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, Z_1)) \text{ and}$$
$$\exists \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{RF}(\mathsf{ctr}'))^\top + \mathbf{K}^\top[\mathbf{m}]_1$$

Since the view of the adversary is independent of ctr, we have

$$\mathbf{Adv}_5 = \frac{\mathbf{Adv}_4}{Q}$$

Game 6: In this game, we can replace $\mathbf{K}$ by $\mathbf{K} + \mathbf{v}(\mathbf{a}^\perp)^\top$ for $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^\ell$. Also, we replace $\{\mathbf{RF}(i) : i \in [1, Q], i \neq \mathsf{ctr}\}$ by $\{\mathbf{RF}(i) + \mathbf{w}_i : i \in [1, Q], i \neq \tilde{\mathsf{ctr}}\}$, for $\mathbf{w}_i \xleftarrow{\$} \mathbb{Z}_p^{2k}$ and $i \neq \mathsf{ctr}$. So, in each i-th query, where $i \neq \mathsf{ctr}$, we compute

$$[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp\mathbf{v}^\top)[\mathbf{m}_i]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{RF}(i) + \mathbf{w}_i)^\top[\mathbf{t}]_1$$

Also, for $\tilde{\mathsf{ctr}}$-th query for the message $[\mathbf{m}_{\tilde{\mathsf{ctr}}}]_1$, we compute

$$[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp\mathbf{v}^\top)[\mathbf{m}_{\tilde{\mathsf{ctr}}}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{RF}(\tilde{\mathsf{ctr}}) + \mathbf{w}_i)^\top[\mathbf{t}]_1$$

So, $\mathcal{A}$ must compute the following:

$$[\mathbf{u}_1^*]_1 = \mathbf{K}_0^\top[\mathbf{t}^*]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp\mathbf{v}^\top)[\mathbf{m}^*]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{RF}(\tilde{\mathsf{ctr}}) + \mathbf{w}_i)^\top[\mathbf{t}^*]_1$$

Since $\mathbf{m}^* \neq [\mathbf{m}_{\tilde{\mathsf{ctr}}}]_R$ (in different classes) by definition of the security game, we can argue $\mathbf{v}^\top\mathbf{m}^*$ and $\mathbf{v}^\top\mathbf{m}_{\tilde{\mathsf{ctr}}}$ are two independent values, uniformly random over $\mathbb{G}_1$. So, $\mathcal{A}$ only can guess it with probability of $\frac{1}{p}$. So, we have

$$\mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{SPS-EQ}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{KerMDH}}_{\mathsf{BG}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{core}}_{\mathsf{BG}}(\mathcal{B}_1) + \frac{2Q}{p}$$

□

E Security proofs of our ABC scheme

Theorem 5. Scheme 1 is correct.

Follows by inspection

Theorem 6. If the q-co-DL assumption holds, the ZKPoK's have perfect ZK, SCDS is sound, and SPS-EQ is EUF-CMA secure, then Scheme 1 is unforgeable.

In the proof of unforgeability we distinguish whether the adversary wins the game by forging a signature, breaking the opening soundness of the commitment scheme or computing a discrete logarithm. The proof of unforgeability follows almost verbatim the strategy in [FHS19] with modifications to take care of disjoint sets.

Proof. We first introduce the following syntactic changes to the experiment, which allows us to distinguish forgeries: (1) We include the value R in the credential cred output by Obtain. (2) When the adversary makes a valid call to $\mathcal{O}_{\mathsf{Issue}}$ the experiment receives the values C, R and produces a signature σ; instead of appending $\perp$ to the list $\mathtt{CRED}$, the oracle now appends $((C, R), \sigma, \perp, \perp)$. Note the adversary's view in the experiment remains unchanged.

Assume that an efficient adversary $\mathcal{A}$ wins the unforgeability game with non-negligible probability and let $((C_1^*, C_2^*, C_3^*), \sigma^*)$ be the message-signature pair it uses and wit^* be the witness for an attribute set $\mathcal{S}^* \not\subseteq \mathtt{ATTR}[j]$, or $\underline{\mathsf{wit}}^*$ be the witness for an attribute set $\mathcal{D}^* \subseteq \mathtt{ATTR}[j]$ for all j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. We distinguish the following cases:

Type 1: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} \neq [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, *, *) = \mathtt{CRED}[j]$ for all issuance indices j (*i.e.,* $\mathtt{OWNR}[j] \in \mathtt{HU} \cup \mathtt{CU}$). The pair $((C_1^*, C_2^*, C_3^*), \sigma^*)$ is a signature forgery and using $\mathcal{A}$ we construct and adversary $\mathcal{B}$ that breaks the EUF-CMA security of the SPS-EQ scheme.

Type 2: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} = [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, *, *) = \mathtt{CRED}[j]$ for some index j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. Since $\mathcal{A}$ wins if (1) $\mathcal{S} \not\subseteq \mathtt{ATTR}[j]$ or (2) $\mathcal{D} \subseteq \mathtt{ATTR}[j]$, it must have broken the soundness of the set-commitment scheme SCDS.

Type 3: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} = [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, r, O) = \mathtt{CRED}[j]$ for some index j with $\mathtt{OWNR}[j] \in \mathtt{HU}$. Then we use $\mathcal{A}$ to break q-co-DL.

Type 1. In this case $\mathcal{B}$ interacts with a challenger $\mathcal{C}$ in the EUF-CMA game of SPS-EQ and simulates the ABC-unforgeability game for $\mathcal{A}$. The challenger $\mathcal{C}$ runs $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{crs})$ and gives opk to $\mathcal{B}$. Then $\mathcal{B}$ selects $a \xleftarrow{\$} \mathbb{Z}_p$, defines $\mathsf{scds}_{\mathsf{pp}}$ and sets $(\mathsf{osk}, \mathsf{opk}) \leftarrow (a, \mathsf{pk})$. Then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{opk})$ and simulates the environment and the oracles. All oracles are executed as in the real game, except the following which use the signing oracle instead of the signing key osk.

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{X})$: $\mathcal{B}$ computes $(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{ek}, \mathcal{X}; \mathsf{usk})$, picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and then queries its oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ on $(C, r \cdot C, P)$ to obtain σ. $\mathcal{B}$ appends $(i, ((C, r \cdot C), \sigma, r, O), \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: Instead of signing (C, R, P), $\mathcal{B}$ obtains the signature σ from $\mathcal{C}$'s signing oracle. If successful, $\mathcal{B}$ appends $(i, ((C, R), \sigma, \perp, \perp), \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$, then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. If $\mathcal{A}$ produces a valid showing using a credential $((C_1^*, C_2^*, C_3^*), \sigma^*)$, then $\mathcal{B}$ rewinds $\mathcal{A}$ to the step after sending the commitments and restarts $\mathcal{A}$ with a new challenge $e' \neq e$. Then $\mathcal{B}$ performs a Schnorr-like knowledge extraction to obtain μ. If there is a credential $\perp \neq ((C', R'), \sigma', *, *) \in \mathtt{CRED}$ such that $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$ then $\mathcal{B}$ aborts (as the forgery is not of type 1). Otherwise, $\mathcal{B}$ has never queried a signature for class $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}}$ and outputs $((C_1^*, C_2^*, C_3^*), \sigma*)$ as a forgery. $\mathcal{B}$ thus breaks the EUF-CMA security of SPS-EQ.

Type 2. Adversary $\mathcal{B}$ interacts with the challenger $\mathcal{C}$ in the soundness game for SCDS for some $q \geq 0$. First, $\mathcal{C}$ generates set-commitment parameters $\mathsf{scds}_{\mathsf{pp}} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i \in [q]})$ with $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$ and sends $\mathsf{scds}_{\mathsf{pp}}$ to $\mathcal{B}$. $\mathcal{B}$ generates a key pair $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{crs})$ and runs $\mathcal{A}(\mathsf{opk})$, simulating the oracles. All oracles are as in the real game, except $\mathcal{O}_{\mathsf{Obtain}}$ in which $\mathcal{O}_{\mathsf{Issue}}$ is simulated as:

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: $\mathcal{B}$ runs $\mathcal{A}$ twice to extract usk and sets $\mathtt{USK}[i] \leftarrow \mathsf{usk}$.

When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. Assume $\mathcal{A}$ produces a valid showing using $((C_1^*, C_2^*, C_3^*), \sigma^*)$ and a witness wit^* for the attribute set

$\mathcal{S}^*$, or a valid witness $\underline{\mathsf{wit}}^*$ for the attribute set $\mathcal{D}^*$ such that $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j]$ or $\mathcal{D}^* \subseteq \mathtt{ATTR}[j]$ for all j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. Then $\mathcal{B}$ rewinds $\mathcal{A}$ to the step after sending the commitments and restarts $\mathcal{A}$ with a new challenge $e_1' \neq e_1$. $\mathcal{B}$ can then perform a knowledge extraction to obtain μ such that $C_3^* = \mu P$. Let $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$: if there is no credential $\perp = ((C', R'), *, *, *) \in \mathtt{CRED}$ then $\mathcal{B}$ aborts as the forgery was of type 1. Otherwise, let j^* be such that $((C', R'), *, *, *) = \mathtt{CRED}[j^*]$. If $\mathtt{OWNR}[j^*] \in \mathtt{HU}$ then $\mathcal{B}$ aborts as the forgery is of Type 3. Else we have $\mathtt{OWNR}[j^*] \in \mathtt{CU}$ and $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j^*]$ or $\mathcal{D}^* \subseteq \mathtt{ATTR}[j^*]$. If for some $a' \in \mathtt{ATTR}[j^*] : a'P = aP$ then $\mathcal{B}$ sets $O^* \leftarrow (1, a')$. Else, $\mathcal{B}$ sets $O^* \leftarrow (0, \mu \cdot \mathtt{USK}[\mathtt{OWNR}[j^*]])$. $\mathcal{B}$ outputs $(C_1^*, \mathtt{ATTR}[j^*], O^*, \mathcal{S}^*, \mathsf{wit}^*)$ which satisfies $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j^*] \neq \perp$ and $\mathsf{VerifySS}(\mathsf{pp}, C_1^*, \mathcal{S}^*, \mathsf{wit}^*) = 1$ or $\mathcal{B}$ outputs $(C_1^*, \mathtt{ATTR}[j^*], O^*, \mathcal{D}^*, \underline{\mathsf{wit}}^*)$ which satisfies $\mathcal{D}^* \subseteq \mathtt{ATTR}[j^*] \neq \perp$ and $\mathsf{VerifyDS}(\mathsf{pp}, C_1^*, \mathcal{D}^*, \underline{\mathsf{wit}}^*) = 1$.

$\mathcal{B}$'s output thus breaks the subset- or disjoint-set soundness of SCDS.

Type 3. In this case we assume $\mathcal{A}$ can produce a forgery by computing a discrete log. We proceed via a sequence of games which are indistinguishable under q-co-DL. We denote an adversary succeeding to win **Game** i by S_i.
Game 0: The original game, which only outputs 1 if the forgery is of Type 3. **Game 1**: As **Game 0**, except for the following oracles:

$\mathcal{O}_{\mathsf{Obtlss}}(i, \mathcal{S})$: As in **Game 0**, except that the experiment aborts if the set-commitment trapdoor is contained in $\mathcal{S}$.
$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: As in **Game 0**, except that the experiment aborts if the set-commitment trapdoor is contained in $\mathcal{S}$.

Game 0 → Game 1: If $\mathcal{A}$ queries either set $\mathcal{S}, \mathcal{D}$ with $s \in \mathcal{S}$ or $d \in \mathcal{D}$ to one of the two oracles, then this breaks the q-co-DL assumption for $q = s$ and $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. Denoting the advantage of solving the q-co-DL by $\epsilon_{qDL}(\lambda)$, we have

$$|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_{qDL}(\lambda).$$

Game 2: As **Game 1**, with the difference that the oracle $\mathcal{O}_{\mathsf{Show}}$ is run as follows

$\mathcal{O}_{\mathsf{Show}}(j, \mathcal{S})$: As in **Game 0**, but the freshness is simulated by leveraging the fact that the environment has access to the TSetup algorithm. The proof of knowledge can be run as normal, but given access to td the elements of the Pederson commitment can be changed.

Game 1 → Game 2: By the perfect zero-knowledge property we have

$$\Pr[S_1] = \Pr[S_2].$$

Game 3: As **Game 2**, except that the oracle $\mathcal{O}_{\mathtt{HU}}$ is run as follows:

$\mathcal{O}_{\mathtt{HU}}$: As in **Game 1**, but when executing $\mathsf{UsrKGen}(\mathsf{opk})$, the experiment draws $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p$ instead of $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$ and aborts if $\mathsf{usk} = 0$.

Game 2 → Game 3: Denoting by q_u the number of queries to $\mathcal{O}_{\mathtt{HU}}$, we have

$$|\Pr[S_2] - \Pr[S_3]| \leq \frac{q_u}{p}.$$

Game 4: As **Game 3**, except that when $\mathcal{A}$ eventually delivers a valid showing, the experiment rewinds $\mathcal{A}$ to the point before the commitments are sent, issues a new challenge and extracts a witness (r, σ). If the extractor fails, we abort.
Game 3 → Game 4: The success probability in **Game 4** is the same as in **Game 3**, unless the extraction fails, *i.e.,* using knowledge soundness, we have

$$|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{ks}(\lambda).$$

Game 5: As **Game 4**, except that we pick and index $k \xleftarrow{\$} [q_o]$, where q_o is the number of queries to $\mathcal{O}_{\mathsf{Obtlss}}$. Intuitively, this is the environment guessing that the adversary will use the kth issued credential in its Type 3 forgery.

The extracted witness is such that $w = (r, \mu) \in (\mathbb{Z}_p^*)^2$, and $C_2^* = rC_1^*$ and $C_3 = \mu P$. If the credential $((C', R'), \sigma', r', O') \leftarrow \mathtt{CRED}[k]$ is such that $(C', R', P') \neq \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$ then the experiment aborts. We further abort if the adversary wants to corrupt the owner of the kth credential and adapt $\mathcal{O}_{\mathtt{CU}}$ as follows:

$\mathcal{O}_{\mathtt{CU}}(i)$: As in **Game 0**, except that the experiment aborts when $i = \mathtt{OWNR}[k]$.

Game 4 → Game 5: When the forgery is of Type 3 then there exists some j s.t. for $\mathtt{CRED}[j] = ((C', R'), \sigma', r', O')$ we have $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$; moreover, $\mathtt{OWNR}[j] \in \mathtt{HU}$. With probability $\frac{1}{q_o}$ we have $k = j$, in which case the experiment does not abort, *i.e.,* we have

$$\Pr[S_5] \geq \frac{1}{q_o} \Pr[S_4]$$

We will now show that $\Pr[S_5] \leq \epsilon_{DL}(\lambda)$, where $\epsilon_{DL}(\lambda)$ is the advantage of solving the DLP. $\mathcal{B}$ plays the role of the challenger for $\mathcal{A}$ in **Game 5** and obtains a $\mathbb{G}_1$-DLP instance (BG, xP). $\mathcal{B}$ generates a key pair $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OrgKGen}(\mathsf{crs})$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{opk})$ and simulates the oracles as in **Game 5**, except for $\mathcal{O}_{\mathsf{Obtlss}}$, whose simulation is as follows:

$\mathcal{O}_{\mathsf{Obtlss}}(i, \mathcal{S})$: Let this be the jth query. $\mathcal{B}$ first computes $C \leftarrow \mathtt{USK}[i] \cdot \mathsf{Ch}_{\mathcal{X}}(a) \cdot xP (= x \cdot C), O = (O, \mathtt{USK}[i])$ and appends $\mathsf{cred} = ((C, R)\sigma, \bot, O)$ to $\mathtt{CRED}$. Otherwise **B** proceeds as in **Game 5**.

Note that since **Game 2**, the third component r of the credential is not required to simulate $\mathcal{O}_{\mathsf{Show}}$ queries. When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$ then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. If $\mathcal{A}$ wins **Game 5** using (C_1^*, C_2^*, C_3^*) and conducting the proof of knowledge on the freshness, then $\mathcal{B}$ can rewind $\mathcal{A}$ and extract a witness $w = (r', \mu) \in (\mathbb{Z}_p^*)^3$ such that $C_2^* = r' C_1^*$ and $C_3^* = \mu P$. Further, we have that $((C', R'), \sigma', \bot, O') = \mathtt{CRED}[k]$. In the end, $\mathcal{B}$ outputs r' as a solution to the DLP in $\mathbb{G}_1$. We thus have

$$\Pr[S_5] \leq \epsilon_{DL}(\lambda)$$

Collecting the success probabilities, we have $\Pr[S_0] \leq q_o \cdot \epsilon_{DL}(\lambda) + \epsilon_{ks}(\lambda) + \frac{q_u}{p} + \epsilon_{qDL}(\lambda)$ where $q = t$ and q_o and q_u and the number of queries to $\mathcal{O}_{\mathsf{Obtlss}}$ and $\mathcal{O}_{\mathtt{HU}}$ respectively. □

Theorem 7. If the DDH assumption holds, the ZKPoK's have perfect ZK, and the SPS-EQ perfectly adapts signatures, then Scheme 1 is anonymous.

The following proof is an adaptation (most of it verbatim) of the one given in [FHS19]. The only different is that since we use a CRS and manage a slightly different definition for perfect adaption, we need to adjust the previous proof for this new setting. For ease of exposition we only consider selective disclosures showings in the proof, but the adecuation for NAND showings follows directly. As in [FHS19], the proof proceeds by defining a sequence of indistinguishable games in the last of which the answers of $\mathcal{O}_{LoR}$ are independent of the bit b.

Proof. We assume that the adversary $\mathcal{A}$ will call $\mathcal{O}_{LoR}$ for some $(j_0, j_1, \mathcal{S})$ with both $\mathtt{OWNR}[j_0], \mathtt{OWNR}[j_1] \in \mathtt{HU}$. This is w.l.o.g. as otherwise the bit b is perfectly hidden from $\mathcal{A}$. Henceforth, we denote the event that the adversary wins Game i by S_i.
Game 0: The original anonymity game as given in Section 6.1.

Game 1: As **Game 0**, except we replace Setup with TSetup.

Game 0 → Game 1: The adversary's view does not change so we have

$$\Pr[S_0] = \Pr[S_1].$$

Game 2: As **Game 1**, except that the experiment runs $\mathcal{O}_{\mathsf{LoR}}$ as follows:

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 1**, but the ZKPoK for (C_1^*, C_2^*, C_3^*) is simulated.

Game 1 → Game 2: By perfect zero-knowledge of Π_2, we have

$$\Pr[S_1] = \Pr[S_2].$$

Game 3: As **Game 2**, except for the following changes. Let q_u be (an upper bound on) the number of queries made to $\mathcal{O}_{\mathtt{HU}}$. At the beginning **Game 2** pick $k \xleftarrow{\$} [q_u]$ (it guesses that the user that owns the j_bth credential is registered at the kth call to $\mathcal{O}_{\mathtt{HU}}$) and runs $\mathcal{O}_{\mathtt{HU}}$, $\mathcal{O}_{\mathtt{CU}}$ and $\mathcal{O}_{\mathsf{LoR}}$ as follows:

$\mathcal{O}_{\mathtt{HU}}(i)$: As in **Game 2**, except if this is the kth call to $\mathcal{O}_{\mathtt{HU}}$ then it additionally defines $i^* \leftarrow i$.

$\mathcal{O}_{\mathtt{CU}}(i, \mathsf{upk})$: If $i \in \mathtt{CU}$ or $i \in I_{\mathsf{LoR}}$, it returns $\perp$ (as in the previous games). If $i = i^*$ then the experiment stops and outputs a random bit $b' \xleftarrow{\$} \{0,1\}$. Otherwise, if $i \in \mathtt{HU}$ it returns user i's usk and credentials and moves i from $\mathtt{HU}$ to $\mathtt{CU}$; and if $i \notin \mathtt{HU} \cup \mathtt{CU}$, it adds i to $\mathtt{CU}$ and sets $\mathtt{UPK}[i] \leftarrow \mathsf{upk}$.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 2**, except that if $i^* \neq \mathtt{OWNR}[j_b]$, the experiment stops outputting $b' \xleftarrow{\$} \{0,1\}$.

Game 2 → Game 3: By assumption, $\mathcal{O}_{\mathsf{LoR}}$ is called at least once with some input $(j_0, j_1, \mathcal{S})$ with $\mathtt{OWNR}[j_0]$, $\mathtt{OWNR}[j_1] \in \mathtt{HU}$. If $i^* = \mathtt{OWNR}[j_b]$ then $\mathcal{O}_{\mathsf{LoR}}$ does not abort and neither does $\mathcal{O}_{\mathtt{CU}}$ (it cannot have been called on $\mathtt{OWNR}[j_b]$ before that call to $\mathcal{O}_{\mathsf{LoR}}$ (otherwise $\mathtt{OWNR}[j_b] \notin \mathtt{HU}$); if called afterwards, it returns $\perp$, since $i^* \in I_{\mathsf{LoR}}$). Since $i^* = \mathtt{OWNR}[j_b]$ with probability $\frac{1}{q_u}$, the probability that the experiment does not abort is at least $\frac{1}{q_u}$, and thus

$$\Pr[S_3] \geq (1 - \tfrac{1}{q_u})\tfrac{1}{2} + \tfrac{1}{q_u} \cdot \Pr[S_2].$$

Game 4: Same as **Game 3**.

Game 3 → Game 4: Let $(\mathsf{BG}, xP_1, yP_1, zP_1)$ be a DDH instance for $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. After initializing the environment, the simulation initializes a list $L \leftarrow \emptyset$. The oracles are simulated as in **Game 3**, except for the subsequent oracles, which are simulated as follows:

$\mathcal{O}_{\mathtt{HU}}(i)$: As in **Game 3**, but if this is the kth call then, besides setting $i^* \leftarrow i$, it sets $\mathtt{USK}[i]\perp$ and $\mathtt{UPK}[i] \leftarrow xP_1$ (which implicitly sets $\mathsf{usk} \leftarrow x$)

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$: As in **Game 3**, except for the computation of the following values if $i = i^*$. Let this be the jth call to this oracle. If $s \notin \mathcal{X}$, it computes C as $C \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot xP_1$ and sets $L[j] \leftarrow \perp$. If $s \in \mathcal{X}$ it picks $\rho \xleftarrow{\$} \mathbb{Z}_p^*$, computes C as $C \leftarrow \rho \cdot xP_1$, sets $L[j] \leftarrow \rho$ and simulates the ZKPoK for upk (by the perfect ZK property of the simulation is perfect). (In both cases C is thus distributed as in the original game.)

$\mathcal{O}_{\mathsf{Show}}(i, \mathcal{S})$: As in **Game 3**, with the difference that if $\mathtt{OWNR}[j] = i^*$ and $s \notin \mathcal{S}$ it computes the witness $\mathsf{wit} \leftarrow \mu\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s) \cdot xP_1$. ($\mathsf{wit}$ is thus distributed as in the original game.)

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 3**, with the following difference. Using self-reducibility of DDH, it picks $s, t \xleftarrow{\$} \mathbb{Z}_p$ and computes $Y' \leftarrow t \cdot yP_1 + sP_1 = y'P_1$ with $y' \leftarrow ty + s$, and $Z' \leftarrow t \cdot zP_1 + s \cdot xP_1 = (t(z - xy) + xy')P_1$.(If $z \neq xy$ then Y' and Z' are independently random; otherwise $Z' = y'X$.) It performs the showing using the following values (implicitly setting $\mu \leftarrow y'$):

- If $s \notin \mathtt{ATTR}[j_b]$: $C_1 \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)Z'$ and $\mathsf{wit} \leftarrow \mathsf{Ch}_{\mathcal{S}}(s)^{-1}C_1$
- If $s \in \mathtt{ATTR}[j_b]$ and $s \notin \mathcal{S}$: $C_1 \leftarrow \rho Z'$ with $\rho \leftarrow L[j_b]$ and $\mathsf{wit} \leftarrow \mathsf{Ch}_{\mathcal{S}}(s)^{-1}C_1$;
- If $s \in \mathcal{S}$: $c_1 \leftarrow \rho Z'$ with $\rho \leftarrow L[j_b]$ and $\mathsf{wit} \leftarrow \perp$;

Apart from an error event happening with negligible probability, we have simulated **Game 3** if the DDH instance was "real" and **Game 4** otherwise. If $xP_1 = 0_{\mathbb{G}_1}$, or if during the simulation of $\mathcal{O}_{\mathsf{LoR}}$ it occurs that $Y' = 0_{\mathbb{G}_1}$ or $Z' = 0_{\mathbb{G}_1}$ then the distribution of values is not as in one of the two games. Otherwise, we have implicitly set $\mathsf{usk} \leftarrow x$ and $\mu \leftarrow y'$ (for a fresh value y' at every call of $\mathcal{O}_{\mathsf{LoR}}$). In case of a DDH instance, we have (depending on the case) $C1 \leftarrow \mathsf{usk}\mu\mathsf{Ch}_{\mathcal{X}}(s) \cdot P_1$ (or $C_1 = \rho \cdot x\mu \cdot P_1 = \mu \cdot C$). Letting $\epsilon_{DDH}(\lambda)$ denote the advantage of solving the DDH problem and q_l the number of queries to the $\mathcal{O}_{\mathsf{LoR}}$, we have

$$|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{DDH}(\lambda) + (1 + 2q_l)\tfrac{1}{p}.$$

Game 5: Same as **Game 4**.

Game 4 → Game 5: Let $(\mathsf{BG}, xP_1, yP_1, zP_1)$ be a DDH instance for $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. After initializing the environment, the simulation initializes a list $L \leftarrow \emptyset$. The oracles are simulated as in **Game 4**, except for the subsequent oracles, which are simulated as follows:

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$: As in **Game 4**, except for the computation of the following values if $i = i^*$.Let this be the jth call to this oracle. It first picks $u \xleftarrow{\$} \mathbb{Z}_p$ and sets $X' \leftarrow xP_1 + u \cdot P_1$ and $L[j] \leftarrow u$. If $s \notin \mathcal{X}$, it computes $C \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot \mathtt{USK}[i]P_1$ and $R \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot \mathtt{USK}[i]X'$. If $s \in \mathcal{X}$, it picks $\rho \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $C \leftarrow \rho P_1$ and $R \leftarrow \rho X'$.In both cases it sets $r \leftarrow \perp$ (r is implicitly set to $r \leftarrow x' := x + u$ and C and $R = rC$ are distributed as in the original game; unless $X' = 0_{\mathbb{G}_1}$). Note that, since the ZKPoK in $\mathcal{O}_{\mathsf{Show}}$ is simulated, r is not used anywhere in the game.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 4**, with the difference that it fetches $u \leftarrow L[j_b]$, picks $s, t \xleftarrow{\$} \mathbb{Z}_p$ and recomputes $Y' \leftarrow t \cdot yP_1 + sP_1 = y'P_1$ with $y' \leftarrow ty + s$,and $Z' \leftarrow t \cdot zP_1 + s \cdot xP_1 + ut \cdot yP_1 + us \cdot P_1 = (t(z - xy) + x'y')P_1$. It performs the showing as in the previous simulation (using the new Y', Z' and $\mu \leftarrow y'$).

Apart from an error event happening with negligible probability, we have simulated **Game 4** if the DDH instance was valid and **Game 5** otherwise. If $X' = 0_{\mathbb{G}_1}$ during the simulation of $\mathcal{O}_{\mathsf{Obtain}}$, or if during the simulation of $\mathcal{O}_{\mathsf{LoR}}$ it occurs that $Y' = 0_{\mathbb{G}_1}$ or $Z' = 0_{\mathbb{G}_1}$ then the distribution of values is not as in one of the two games. Otherwise, we have implicitly set $r \leftarrow x'$ (for a fresh value x' at every call of $\mathcal{O}_{\mathsf{Obtain}}$). Letting $\epsilon_{DDH}(\lambda)$ denote the advantage of solving the DDH problem, and q_o and q_l be the number of queries to $\mathcal{O}_{\mathsf{Obtain}}$ and $\mathcal{O}_{\mathsf{LoR}}$, respectively, we get

$$|\Pr[S_4] - \Pr[S_5]| \leq \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\tfrac{1}{p}.$$

In **Game 5**, by definition of perfect adaption the oracle $\mathcal{O}_{\mathsf{LoR}}$ returns a signature that is a random element in the space of signatures conditioned to verify with the shown credential (each generated with fresh independent randomness $\mu \leftarrow y'$, when calling the oracle $\mathcal{O}_{\mathsf{LoR}}$), and with respect to a simulated proof. Hence, the bit b is information-theoretically hidden from $\mathcal{A}$ and we have $\Pr[S_5] = \frac{1}{2}$. Therefore, we have that:

$$\Pr[S_4] \leq \Pr[S_5] + \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\frac{1}{p} = \frac{1}{2} + \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\frac{1}{p},$$
$$\Pr[S_3] \leq \Pr[S_4] + \epsilon_{DDH}(\lambda) + (1 + 2q_l)\frac{1}{p} \leq \frac{1}{2} + 2 \cdot \epsilon_{DDH}(\lambda) + (1 + q_o + 4q_l)\frac{1}{p},$$
$$\Pr[S_2] \leq \frac{1}{2} + q_u \cdot \Pr[S_3] - \frac{1}{2} \cdot q_u \leq \frac{1}{2} + q_u \cdot (2 \cdot \epsilon_{DDH}(\lambda) + (1 + q_o + 4q_l)\frac{1}{p}),$$

where $\Pr[S_2] = \Pr[S_1] = \Pr[S_0]$; q_u, q_o and q_l are the number of queries to $\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathsf{Obtain}}$ and $\mathcal{O}_{\mathsf{LoR}}$, respectively. Assuming security of the ZKPoKs and DDH, the adversary's advantage is thus negligible. □

F Revocation strategies

The revocation approach from [DHS15a] requires the authority to compute and mantain the witness list. We observe that instead of using the accumulator from [ATSM09] one could use the one from [KB21], which has constant size non-membership proofs at the cost of a more expensive setup involving all the pseudonyms (up to the upper bound). If such accumulator is used, to keep most of the computational cost done in $\mathbb{G}_1$ and not in $\mathbb{G}_2$, verification under the revocation model from [DHS15a] would now require the evaluation of four pairings by the verifier and none by the user (before it was two and one respectively) but one less ZKPoK. By doing this, users can compute their non-membership witness more efficiently and the overhead related to the accumulator management is reduced to the minimum. The authority just needs to update the accumulator value every time users are revoked or unrevoked.

In Figure 8 we present the accumulator construction from [KB21] tailored to batch addition and deletions, which simplifies the management.

Another alternative to manage revocation would be to leverage NAND proofs with the following idea. When a credential is issued, users include a pseudonym given by the authority in their credential and then, to prove they are not revoked, given a public list of revocated users they compute a NAND proof for the list with respect to their own credential. However, this requires the revocation list to be kept below the size limit of the set-commitment scheme.

G Expressiveness of our Set-commitment scheme

In this section we elaborate on the expressivity gained by adding a set-commitment supporting proofs of disjoint sets (*i.e.,* NAND showings).

First of all, scenarios considering access control policies can benefit of NAND showings as they allow users, for instance, to prove that they do not belong to a particular business unit within a company. Another example includes applications providing discounts to tourists of a particular region. With the ABC [FHS19] there is no easy way for users to prove that they are not nationals

$\underline{\mathsf{RevAcc.Setup}(1^\lambda, 1^q)}$:
$\mathsf{BG} \xleftarrow{\$}; \mathsf{BGGen}(1^\lambda); \mathsf{td} = s \leftarrow \mathbb{Z}_p^*$
return $(\mathsf{BG}, (s^i P_1, s^i P_2)_{i\in[q]}, \mathsf{td})$

$\underline{\mathsf{RevAcc.Commit}(\mathsf{pp}, \mathcal{X})}$:
if $|\mathcal{X}| > q \vee \exists\, s' \in \mathcal{X} : s'P_1 = sP_1$ **return** $\bot$
else
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{acc}_1 \leftarrow r_1 P_1$; $\mathsf{acc}_2 \leftarrow r_2 \cdot \mathsf{Ch}_{\mathcal{X}}(s) P_1$
return $((\mathsf{acc}_1, \mathsf{acc}_2), (r_1, r_2))$

$\underline{\mathsf{RevAcc.NonMemWit}(\mathsf{pp}, \mathsf{td}, \mathsf{acc}^t, \tau)}$:
return $(\frac{1}{\mathsf{td}+\tau})\mathsf{acc}_2^t$

$\underline{\mathsf{RevAcc.VerifyWit}(\mathsf{pp}, \mathsf{acc}, \tau, \mathsf{wit})}$:
return $e(\mathsf{wit}, (\mathsf{td}+\tau)P_2) = e(\mathsf{acc}, P_2)$

$\underline{\mathsf{RevAcc.Add}(\mathsf{pp}, \mathsf{td}, \mathsf{acc}^t, \mathsf{aux}^t, \mathcal{S})}$:
$r_1', r_2' \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{acc}^{t+1} \leftarrow ((r_1' \cdot \mathsf{Ch}_{\mathcal{S}}(s))\mathsf{acc}_1^t, \frac{r_2'}{\mathsf{Ch}_{\mathcal{S}}(s)}\mathsf{acc}_2^t)$
$\mathsf{aux}^{t+1} \leftarrow (r_1'\mathsf{aux}_1^t, r_2'\mathsf{aux}_2^t)$
return $(\mathsf{acc}^{t+1}, \mathsf{aux}^{t+1})$

$\underline{\mathsf{RevAcc.Del}(\mathsf{pp}, \mathsf{td}, \mathsf{acc}^t, \mathsf{aux}^t, \mathcal{S})}$:
$r_1', r_2' \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{acc}^{t+1} \leftarrow (\frac{r_1'}{\mathsf{Ch}_{\mathcal{S}}(s)}\mathsf{acc}_1^t, (r_2' \cdot \mathsf{Ch}_{\mathcal{S}}(s))\mathsf{acc}_2^t)$
$\mathsf{aux}^{t+1} \leftarrow (r_1'\mathsf{aux}_1^t, r_2'\mathsf{aux}_2^t)$
return $(\mathsf{acc}^{t+1}, \mathsf{aux}^{t+1})$

Fig. 8: Revocation accumulator from [KB21] tailored to batch updates.

of a particular country. Using a NAND proof, any user from Spain can easily prove to the italian authorities that is a tourist computing a NAND proof for the attribute {"residence, Italy"}.

Furthermore, we borrow the following example from [TG20] (Section 6.2) to illustrate how NAND showings can be used to perform interval proofs. Suppose that a users want to prove that they are at least 18 year-old. Assuming the current date is 2 January 2020 and the user's birthday is on 1 January 2002, we can have two redundant attributes {"byear = 2002"}, {"bmth = Jan2002"} for {"bday = 01Jan2002"} in the user's credential so that the verifier can ask for a NAND showing on the attribute set

$$\mathcal{D}=\{\{\text{"byear = 2020"}\}, \ldots, \{\text{"byear = 2003"}\}, \{\text{"bmth = Feb2002"}\}, \ldots, \{\text{"bmth = Dec2002"}\}, \{\text{"bday = 02Jan2002"}\}, \ldots, \{\text{"bday = 31Jan2002"}\}\}$$

Such a proof can only be done in the ABC from [FHS19] encoding predefined statements like {"adult, >18"}.

H Mercurial Signatures

In the following we present the required changes on our scheme to obtain a mercurial signature. It suffices to define the algorithms $\mathsf{ConvertPK}, \mathsf{ConvertSK}$ and $\mathsf{ConvertSig}$ as in shown in Figure 9.

With respect to the security properties, we need to consider now the perfect adaptation of signatures with respect to the key space. As pointed out in [KSD19], perfect adaption is a stronger notion than the one from original class-hiding (which was the one used in earlier works on mercurial signatures [CL19, CL21]). We therefore follow the former approach and work with the definition of perfect adaption. That being said, as in previous constructions of mercurial signatures, ours only satisfies perfect adaption under *honestly* generated keys (otherwise signers could identify adapted signatures).

Theorem 11. The resulting mercurial signature from Figure 9 has perfect adaption of signatures (under honestly generated keys in the honest parameters model) with respect to the key space.

Proof. For all $[\mathbf{m}]_1$ and $\mathsf{pk} = ([\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$, a signature σ=$([\mathbf{u}_1]_1$,$[\mathbf{t}]_1$,Ω_1,$[e_0]_2$, $[e_1]_2$,$E_1)$ generated according to the CRS $([\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [e]_2)$ satisfying the verification algorithm must be of the form: σ=$(\mathbf{K}_0^\top[\mathbf{A}_0]_1 r_1$+$\mathbf{K}^\top[\mathbf{m}]_1$, $[\mathbf{A}_0]_1 r_1$, $[\mathbf{A}_0]s_1$, $[\mathbf{A}_1]d_1^1 - e_1[\mathbf{A}_0]_1 r_1$, $[e_0]_2 r_1$+$[s_1]_2$, $[d_1^1]_2$, $[e_0]_2$, $[e_1]_2$, $E_1)$. A signature output by $\mathsf{ConvertSig}$ has the form σ=$(\rho\mathbf{K}_0^\top[\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$+$\rho\mathbf{K}^\top[\mu\mathbf{m}]_1$, $[\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$, $[\mathbf{A}_0]\alpha(\mu s_1 + \beta s_2)$, $[\mathbf{A}_1]\alpha(\mu d_1^1 + \beta d_1^2) - e_1[\mathbf{A}_0]_1\alpha(\mu r_1 + \beta r_2)$, $\alpha([e_0]_2(\mu r_1 + \beta r_2)$+$\mu[s_1]_2$+$\beta[s_2]_2)$, $\alpha(\mu[d_1^1]_2 + \beta[d_1^2]_2)$, $\alpha[e_0]_2$, $\alpha[e_1]_2$, $\alpha E_1)$, for new independent randomness α, β, μ and ρ so is a random element in the space of all signatures. Furthermore, the signature output by ChgRep is distributed identically to a fresh signature on message $[\mathbf{m}]_1$ output by $\mathsf{Sign}(\mathsf{pp}, \mathsf{ConvertSK}(\mathsf{sk}, \rho), [\mathbf{m}]_1)$. □

Remark. Since our construction requires the use of a tag, to implement the delegatable credentials from [CL19] with our construction, users would be required to store the tags and to randomize them when delegating to another user (*i.e.,* tags need to be randomized and passed along for each delegation level with the corresponding signature). Therefore, users should verify the tag correctness when obtaining a signature but such verification would still not be required for credential presentations.

$\underline{\mathsf{SPS\text{-}EQ.ConvertSK}(\mathsf{sk}, \rho)}$:
parse $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K})$; **return** $(\rho\mathbf{K}_0, \rho\mathbf{K})$

$\underline{\mathsf{SPS\text{-}EQ.ConvertPK}(\mathsf{pk}, \rho)}$:
parse $\mathsf{pk} = ([\mathbf{B}]_2, [\mathbf{C}]_2)$; **return** $(\rho[\mathbf{B}]_2, \rho[\mathbf{C}]_2)$

$\underline{\mathsf{SPS\text{-}EQ.ConvertSig}(\mathsf{crs}, [\mathbf{m}]_1, \sigma, \tau, \mu, \boldsymbol{\rho}, \mathsf{pk})}$:
parse $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
parse $\tau \in \{([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \cup \bot\}$
$\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
check $\mathsf{PVer}(\mathsf{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$
check $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
check $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\mathbf{m}]_1^\top, [\mathbf{C}]_2)$
$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$
$[\mathbf{u}_1']_1 := \boldsymbol{\rho}(\mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1)$
$[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$
for all $i \in \{0, 1\}$
  $[z_i']_2 \leftarrow \alpha[z_i]_2$
  $[\mathbf{a}_i']_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$
  $[d_i']_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$
$\Omega' \leftarrow (([\mathbf{a}_i']_1, [d_i']_2, [z_i']_2)_{i\in\{0,1\}}, \alpha Z_1)$
$\sigma' \leftarrow ([\mathbf{u}_1']_1, [\mathbf{t}']_1, \Omega')$
return $(\mu[\mathbf{m}]_1, \sigma')$

Fig. 9: Our mercurial signature scheme (ConvertSig also changes the message representative).

I Our full ABC scheme

$\underline{\mathsf{ABC.Setup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$; $(\mathsf{sps}_{\mathsf{pp}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ParGen}(1^\lambda; \mathsf{BG})$;
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$; **return** $(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{pp}}, \mathsf{ck})$

$\underline{\mathsf{ABC.TSetup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{scds}_{\mathsf{td}}) \xleftarrow{\$} \mathsf{SCDS.TSetup}(1^\lambda, q)$; $(\mathsf{sps}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{td}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.TParGen}(1^\lambda; \mathsf{BG})$;
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$; $\mathsf{ck}_{\mathsf{td}} \leftarrow r$; **return** $((\mathsf{BG}, \mathsf{scds}_{\mathsf{pp}}, \mathsf{sps}_{\mathsf{pp}}, \mathsf{ck}), (\mathsf{scds}_{\mathsf{td}}, \mathsf{sps}_{\mathsf{td}}, \mathsf{ck}_{\mathsf{td}}))$

$\underline{\mathsf{ABC.OrgKGen}(\mathsf{pp})}$:
return $\mathsf{SPS\text{-}EQ.KGen}(\mathsf{BG}, \mathsf{sps}_{\mathsf{pp}}, 3)$

$\underline{\mathsf{ABC.UsrKGen}(\mathsf{pp})}$:
$\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{upk} \leftarrow \mathsf{usk}P_1$; **return** $(\mathsf{usk}, \mathsf{upk})$

$\mathsf{ABC.Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X})$		$\mathsf{ABC.Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$; $a \leftarrow r_1 P_1$		
$c \leftarrow \mathsf{Commit}(\mathsf{ck}, a, r_2)$	$\xrightarrow{c}$	
$z \leftarrow r_1 + e \cdot \mathsf{usk}$	$\xleftarrow{e}$	$e \xleftarrow{\$} \mathbb{Z}_p^*$
$(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{X}; \mathsf{usk})$	$\xrightarrow{C,\, R,\, z, a, r_2}$	**if** $(zP_1 \neq a + e \cdot \mathsf{upk} \ \vee \ c \neq \mathsf{Commit}(\mathsf{ck}, a, r_2))$
$r_3 \xleftarrow{\$} \mathbb{Z}_p^*$; $R \leftarrow r_3 C$		**return** $\perp$
		if $(e(C, P_2) \neq e(\mathsf{upk}, \mathsf{Ch}_{\mathcal{X}}(s) P_2)$
		$\wedge \ \forall\, x \in \mathcal{X} : xP_1 \neq \mathsf{ek}_1^0)$ **return** $\perp$
	$\xleftarrow{(\sigma, \tau)}$	$(\sigma, \tau) \leftarrow \mathsf{SPS\text{-}EQ.Sign}(\mathsf{sps}_{\mathsf{pp}}, (C, R, P_1), \mathsf{osk})$
check $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps}_{\mathsf{pp}}(C, R, P_1), (\sigma, \tau), \mathsf{opk})$		
return $\mathsf{cred} = (C, (\sigma, \tau), r_3, O)$		

$\mathsf{ABC.Show}(\mathsf{pp}, \mathsf{usk}, (\mathsf{opk}_i)_{i \in [n]}, \mathsf{opk}, \mathcal{S}, \mathcal{D}, \mathsf{cred})$		$\mathsf{ABC.Verify}(\mathsf{pp}, (\mathsf{opk}_i)_{i \in [n]}, \mathcal{S}, \mathcal{D})$
parse $\mathsf{cred} = (C, (\sigma, \tau), r, O)$; $\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$		
if $O = (1, (o_1, o_2))$ **then** $O' = (1, (\mu \cdot o_1, o_2))$		
else $O' = \mu \cdot O$		
$\sigma' \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{sps}_{\mathsf{pp}}, (C, rC, P_1), \sigma, \tau, \mu, \rho, \mathsf{opk})$		
$(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$		
$\mathsf{cred}' \leftarrow (C_1, C_2, C_3, \sigma')$; $\mathsf{opk}' \leftarrow \mathsf{ConvertPK}(\mathsf{opk}, \rho)$		
$\Pi \leftarrow \mathsf{SH.PPrv}((\mathsf{opk}_i)_{i \in [n]}, \mathsf{opk}', \rho)$		
$\mathsf{wit} \leftarrow \mathsf{SCDS.OpenSS}(\mathsf{scds}_{\mathsf{pp}}, \mu C, \mathcal{S}, O')$		
$\underline{\mathsf{wit}} \leftarrow \mathsf{SCDS.OpenDS}(\mathsf{scds}_{\mathsf{pp}}, \mu C, \mathcal{D}, O')$		
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$		
$c_1 \leftarrow \mathsf{Commit}(\mathsf{ck}, a_1, r_2)$; $c_2 \leftarrow \mathsf{Commit}(\mathsf{ck}, a_2, r_4)$		
$\Sigma_1 = (\mathsf{cred}', \Pi, \mathsf{opk}', \mathsf{wit}, \underline{\mathsf{wit}}, c_1, c_2)$	$\xrightarrow{\Sigma_1}$	**parse** $\Sigma_1 = (\mathsf{cred}', \Pi, \mathsf{opk}', \mathsf{wit}, \underline{\mathsf{wit}}, c_1, c_2)$
$\pi_1 \leftarrow \mathsf{SCDS.PoE}(\mathsf{ek}, \mathcal{S}, \tilde{e})$	$\xleftarrow{e, \tilde{e}}$	$e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
$\pi_2 \leftarrow \mathsf{SCDS.PoE}(\mathsf{ek}, \mathcal{D}, \tilde{e})$		**parse** $\mathsf{cred}' = (C_1, C_2, C_3, \sigma)$
$z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$		
$\Sigma_2 = (z_i, a_i, r_i, \pi_i)_{i \in \{1,2\}}$	$\xrightarrow{\Sigma_2}$	**parse** $\Sigma_2 = (z_i, a_i, r_i, \pi_i)_{i \in \{1,2\}}$
		check
		$z_1 C_1 = a_1 + eC_2$; $z_2 P_1 = a_2 + eC_3$
		$c_1 = \mathsf{Commit}(\mathsf{ck}, a_1, r_2)$; $c_2 = \mathsf{Commit}(\mathsf{ck}, a_2, r_4)$
		$\mathsf{SH.PVer}(\mathsf{crs}, (\mathsf{opk}_i)_{i \in [n]}, \mathsf{opk}', \Pi_1)$
		$\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps}_{\mathsf{pp}}, \mathsf{cred}', \mathsf{opk}')$
		$\mathsf{SCDS.VerifySS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{S}, \mathsf{wit}; \pi_1, \tilde{e})$
		$\mathsf{SCDS.VerifyDS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{S}, \underline{\mathsf{wit}}; \pi_2, \tilde{e})$

Fig. 10: Full ABC scheme from Section 7.

J On the integration of the proof of exponentiation

Protocol 2.1 and Proposition 2.2 from [Tha19] consider a general pairing e and a general polynomial $f(X) \in \mathbb{F}_p[X]$. The protocol PoE used in this work relies on other restrictions and therefore it's security is not based on Proposition 2.2. In our case, the pairing function e being used is of Type-III and α is chosen by the verifier meaning that β is also determined by α. Below we argue the security of our PoE protocol considering the use of the PoE in the $\mathsf{VerifySS}$ algorithm of our scheme (a similar reasoning also applies to $\mathsf{VerifyDS}$).

The adversary $\mathcal{A}$ is given α (chosen by the verifier at random) and has to produce witnesses w_1, w_2 and w_3 for a set $\mathcal{S}$, with $\mathsf{Ch}_{\mathcal{S}}(X) = (X + \alpha)q_{\mathcal{S}}(X) + \beta$ and s.t the following holds:

$$(e((s + \alpha)P_1, w_1) + e(\beta P_1, P_2) = e(P_1, w_2)) \wedge (e(w_3, w_2) = e(C, P_2))$$

This means that the w_2 used to verify the first pairing equation is also used to verify the second equation and hence, adds a restriction on the witness that needs to be produced in Proposition 2.2 from [Tha19] (where, unlike here, the KEA assumption is required).

We can assume w.l.o.g that w_2 is of the form $\mathsf{Ch}_{\mathcal{D}}(s)P_2$ for some set $\mathcal{D} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the accumulated set by C. Otherwise, even if the first pairing equation is verified, the second will fail if the *q-co-GSDH* assumption holds.

With the above in mind, we study the existence of $\mathsf{Ch}_{\mathcal{D}}(X)$ that verifies the first pairing equation given that $(X + \alpha)$ and β are fixed for the adversary.

We have three cases: (1) $\mathcal{S} \subseteq \mathcal{D}$, (2) $\mathcal{D} \subset \mathcal{S}$ and (3) $\mathcal{D} \cap \mathcal{S} = \emptyset$.

(1) If $\mathcal{S} \subseteq \mathcal{D}$ and the adversary succeeds in producing the witnesses that pass both verifications, a proof for $\mathcal{D} \subseteq \mathcal{X}$ would also work as a proof for $\mathcal{S} \subseteq \mathcal{X}$ (in which case the adversary would be doing extra computations which are not required and thus such a case is not considered as an attack).

(2) $\mathcal{D} \subset \mathcal{S}$: We assume that α and $\alpha + 1$ do not belong to $\mathcal{S}$. We have that:

$$\mathsf{Ch}_{\mathcal{S}}(X) = (X + \alpha)q_{\mathsf{Ch}_{\mathcal{S}}}(X) + \beta \tag{1}$$
$$\mathsf{Ch}_{\mathcal{D}}(X) = (X + \alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X) + \beta \tag{2}$$

We deduce that $\beta = \mathsf{Ch}_{\mathcal{D}}(X) - (X + \alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X)$ then we obtain

$$\mathsf{Ch}_{\mathcal{S}}(X) = (X + \alpha)q_{\mathsf{Ch}_{\mathcal{S}}}(X) + \mathsf{Ch}_{\mathcal{D}}(X) - (X + \alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X)$$

Moreover, we have $\mathsf{Ch}_{\mathcal{S}}(X) = \mathsf{Ch}_{\mathcal{D}}(X)Q(X)$ and so we get that

$$\mathsf{Ch}_{\mathcal{D}}(X)(Q(X) - 1) = (X + \alpha)(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X))$$

Since $\alpha \notin \mathsf{Ch}_{\mathcal{D}}(X)$, the terms $(X+\alpha)$ and $(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X))$ have to divide $Q(X) - 1$ and $\mathsf{Ch}_{\mathcal{D}}(X)$ respectively. Therefore, we have:

$$(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X)) = \mathsf{Ch}_{\mathcal{D}}(X)B$$
$$Q(X) - 1 = (X + \alpha)B$$

From the first equation we get that $deg(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X)) \leq deg(q_{\mathsf{Ch}_{\mathcal{S}}}(X)) = deg(\mathsf{Ch}_{\mathcal{S}}(X)) - 1$ (which follows from (1)). This means that $deg(\mathsf{Ch}_{\mathcal{D}}(X)) \leq deg(\mathsf{Ch}_{\mathcal{S}}(X)) - 1$. Looking at the second equation, we recall that $\mathsf{Ch}_{\mathcal{S}}(X) = \mathsf{Ch}_{\mathcal{D}}(X)Q(X)$. Since $\mathsf{Ch}_{\mathcal{S}}(X)$ and $\mathsf{Ch}_{\mathcal{D}}(X)$ are irreducible polynomials we can deduce that $B = 1$. Hence $Q(X) = (X + (\alpha + 1))$ and $(X + (\alpha + 1))$ is a factor of $\mathsf{Ch}_{\mathcal{S}}(X)$, which contradicts the assumption that $\alpha + 1 \notin \mathcal{S}$. We conclude that no such set $\mathcal{D}$ exists.

(3) $\mathcal{D} \cap \mathcal{S} = \emptyset$: In this case the adversary needs to produce a subset $\mathcal{D} \subseteq \mathcal{X}$ for which the following holds: $\mathsf{Ch}_{\mathcal{D}}(X) = (X + \alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X) + \beta$. In such case, looking at the first pairing equation we have that:

$$e((s + \alpha)P_1, w_1) + e(\beta P_1, P_2) = e(P_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2)$$
$$e(P_1, (s + \alpha)w_1) = e(P_1, (\mathsf{Ch}_{\mathcal{D}}(s) - \beta)P_2)$$

which means that $(s+\alpha)w_1 = (\mathsf{Ch}_{\mathcal{D}}(s) - \beta)P_2$. We show that if such a w_1 exists then we can build and adversary that breaks the *q-co-SDH* assumption. Therefore, we assume that $f(X) = \mathsf{Ch}_{\mathcal{D}}(X) - \beta$ does not divide $(X+\alpha)$. Since $f(X)$ and $(X+\alpha)$ are relatively prime we can compute polynomials $h_1(X), h_2(X)$ such that

$$f(X)h_1(X) + (X+\alpha)h_2(X) = 1$$

Set $w_1^* := h_1(s)w_1 + h_2(s)P_2$. Then $((s+\alpha))w_1^* = P_2$ and we have a pair (α, w_1^*) which breaks the *q-co-SDH* in $\mathbb{G}_2$.