This is the full version of the extended abstract which appears in
Proceedings of the 13th International Conference on Security and Cryptography for Networks (SCN '22)
(12–14 September 2022, Amalfi, Italy)
Clemente Galdi and Stanislaw Jarecki Eds., LNCS 13409, pp. 411-434, Springer-Verlag 2022

Traceable Constant-Size Multi-Authority Credentials

Chloé Hébant[1] and David Pointcheval[2]

[1] Cosmian, Paris, France – chloe.hebant@cosmian.com
[2] DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France

Abstract Many attribute-based anonymous credential (ABC) schemes have been proposed allowing a user to prove the possession of some attributes, anonymously. They became more and more practical with, for the most recent papers, a constant-size credential to show a subset of attributes issued by a unique credential issuer. However, proving possession of attributes coming from K different credential issuers usually requires K independent credentials to be shown. Only attribute-based credential schemes from aggregate signatures can overcome this issue.
In this paper, we propose new ABC schemes from aggregate signatures with randomizable tags. We consider malicious credential issuers, with adaptive corruptions and collusions with malicious users. Whereas our constructions only support selective disclosures of attributes, to remain compact, our approach significantly improves the complexity in both time and memory of the showing of multiple attributes: for the first time, the cost for the prover is (almost) independent of the number of attributes *and* the number of credential issuers. Whereas anonymous credentials require privacy of the user, we also propose the first schemes allowing traceability by a specific tracing authority.

1 Introduction

In an anonymous credential scheme, a user asks to an organization (a credential issuer) a credential on an attribute, so that he can later claim its possession, even multiple times, but in an anonymous and unlinkable way.

Usually, a credential on one attribute is not enough and the user needs credentials on multiple attributes. Hence, the interest of an attribute-based anonymous credential scheme (ABC in short): depending on the construction, the user receives one credential per attribute or directly for a set of attributes. One goal is to be able to express relations between attributes (or at least selective disclosure), with *one showing*. As different attributes may have different meanings (e.g. a university delivers a diploma while a city hall delivers a birth certification), there should be several credential issuers. Besides multiple credential issuers, it can be useful to have a multi-show credential system to allow a user to prove an arbitrary number of times one credential still without loosing anonymity. For that, the showings are required to be unlinkable to each other.

Classically, a credential is a signature by the credential issuer of the attribute with the public key of the user. The latter is thus the only one able to prove the ownership with an interactive zero-knowledge (ZK) proof of knowledge of the secret key. Anonymity is provided by the probabilistic encryption of the signature. As many signature schemes with various interesting properties have been proposed, many ABC schemes have been designed with quite different approaches. We can gather them into two families: the ABC schemes where a credential is obtained on a set of attributes and then, according to the properties of the signature, it is possible either to prove the knowledge of a subset of the attributes (CL-signatures [CL03, CL04], blind signatures [BL13, FHS15]), or to modify some of the attributes to default values (sanitizable signatures [CL13]), or simply to remove them (unlinkable redactable signatures [CDHK15, San20], SPS-EQ with set commitments [FHS19]); and the ABC schemes where the user receives one credential per attribute and then combines them (aggregate signatures [CL11]). In the former family, whereas it is possible to efficiently show a subset of attributes issued in a unique credential, showing attributes coming from K different credential issuers requires K independent credentials to be proven. On the other hand, with aggregate signatures, credentials on different attributes can be combined together even if they have been issued by different credential issuers. This leads to more compact schemes and this paper follows this latter approach.

Moreover, except some constructions based on blind signatures where the credentials can be shown only once, all ABC schemes allow multi-shows. To this aim, they exploit advanced

properties of the signature scheme, with randomizability for anonymity and unlinkability of the showings. Before these properties, one had to use encryption for anonymity and complex zero-knowledge proofs for unforgeability.

1.1 Our Contributions

Our goal is to obtain a compact ABC system with a compact-size credential allowing different credential issuers. Our first contribution is then the formal definition of the scheme which supports possibly malicious credential issuers.

Following the path of anonymous credentials from aggregate signatures [CL11] and inspired by the definition of linearly homomorphic signatures, our second contribution is the formalization of an (aggregate) signature scheme with randomizable tags (ART-Sign). It comes with a practical construction based on a signature scheme of Hébant *et al.* [HPP19, Appendix C.5]. With such a primitive, two signatures of different messages under different keys can be aggregated only if they are associated to the same tag. In our case, tags will eventually be like pseudonyms, but with some properties which make them ephemeral (hence EphemerId scheme) and randomizable. After randomization, while they are still associated to the same user, they will be unlinkable. This will provide anonymity.

The EphemerId scheme provides ephemeral keys to users, that will allow anonymous authentication. Public keys being randomizable, still for a same secret key, multiple authentications will remain unlinkable. In addition, these public keys will be used as (randomizable) tags with the above ART-Sign scheme when the credential issuer signs an attribute. Thanks to aggregation, multiple credentials for multiple attributes and from multiple credential issuers but under the same tag, and thus for the same user, can be combined into a unique compact (constant-size) credential.

We achieve the optimal goal of constant-size multi-show credentials even for multiple attributes from multiple credential issuers and we stress that aggregation can be done on-the-fly, for any selection of attributes issued by multiple credential issuers: our scheme allows multi-show of any selective disclosure of attributes. About security, whereas there exists a scheme proven in the universal composability (UC) framework [CDHK15], for our constructions, we consider a game-based security model for ABC inspired from [FHS19]. As we support different credentials issuers, we additionally consider adaptive corruptions of both credential issuers and users. However, the keys need to be honestly generated, thus our proofs hold in the certified key setting. This is quite realistic, as this is enough to wait for a valid proof of knowledge of the secret key before certifying the public key. As most of the recent ABC schemes, our constructions rely on signature schemes proven in the bilinear generic group model.

Our last contribution is traceability, in the same vein as group signatures: whereas showings are anonymous, a tracing authority owns a tracing key for being able to link a credential to its owner. In such a case, we also consider malicious tracing authorities, with the non-frameability guarantee. As in [CL13] we thus define trace and judge algorithms to trace the defrauder and prove its identity to a judge. This excludes malicious behavior of the tracing authority. Very few papers deal with traceability: the first one [CL13] exploits sanitizable signatures, where the sanitizer can be traced back, but a closer look shows privacy weaknesses (see appendix A) and a more recent one [KL16] that has thereafter been broken [Ver17]. Our scheme is thus the first traceable attribute-based anonymous credential scheme.

1.2 Related Work

The most recent papers on attribute-based anonymous credential schemes are [FHS19, San20]. The former proposes the first constant-size credential to prove k-of-N attributes, with computational complexity in $O(N-k)$ for the prover and in $O(k)$ for the verifier. However, it only works for one credential issuer ($K=1$). The latter one improves this result enabling multiple

Scheme	P	T	k-of-N attributes from $K = 1$ credential issuer			
			|CI key| $\mathbb{G}_1, \mathbb{G}_2$	|Show| $\mathbb{G}_1, \mathbb{G}_2, (\mathbb{G}_T), \mathbb{Z}_p$	Prover exp., pairings	Verifier exp., pairings
[CL11]	s	✗	$\mathbf{1}, \mathbf{1}$	$16, 2, (4), 7$	$16\mathbb{G}_1 + 2\mathbb{G}_2 + 10\mathbb{G}_T$, $18 + k$	$12\mathbb{G}_1 + 20\mathbb{G}_T$, $18 + k$
[FHS19]	s	✗	$0, N$	$8, 1, 2$	$9\mathbb{G}_1 + 1\mathbb{G}_2, 0$	$4\mathbb{G}_1, k + 4$
[San20]	**r**	✗	$0, 2N + 1$	$2, 2, (1), 2$	$(2(N-k)+2)\mathbb{G}_1 + 2\mathbb{G}_2, 1$	$(k+1)\mathbb{G}_1 + 1\mathbb{G}_T, 5$
Sec. 5.2	s	✓	$0, 2k + 3$	$\mathbf{3}, \mathbf{0}, \mathbf{1}$	$\mathbf{6}\mathbb{G}_\mathbf{1}, \mathbf{0}$	$\mathbf{4}\mathbb{G}_\mathbf{1} + \mathbf{k}\mathbb{G}_\mathbf{2}, \mathbf{3}$
App. D	s	✓	$0, 2N + 2$	$\mathbf{3}, \mathbf{0}, \mathbf{1}$	$\mathbf{6}\mathbb{G}_\mathbf{1}, \mathbf{0}$	$\mathbf{4}\mathbb{G}_\mathbf{1} + \mathbf{2N}\mathbb{G}_\mathbf{2}, \mathbf{3}$

Scheme	$k = 1$-of-N attribute from K credential issuers			
	|CI key| $\mathbb{G}_1, \mathbb{G}_2$	|Show| $\mathbb{G}_1, \mathbb{G}_2, (\mathbb{G}_T), \mathbb{Z}_p$	Prover exp., pairings	Verifier exp., pairings
[CL11]	$\mathbf{K} \times (\mathbf{1}, \mathbf{1})$	$16, 2, (4), 7$	$16\mathbb{G}_1 + 2\mathbb{G}_2 + 10\mathbb{G}_T$, $18 + k$	$12\mathbb{G}_1 + 20\mathbb{G}_T$, $18 + k$
[FHS19]	$K \times (0, N)$	$K \times (8, 1, 2)$	$K \times (9\mathbb{G}_1 + 1\mathbb{G}_2, 0)$	$K \times (4\mathbb{G}_1, k + 4)$
[San20]	$K \times (0, 2N + 1)$	$K \times (2, 2, (1), 2)$	$K \times ((2(N-k)+2)\mathbb{G}_1 + 2\mathbb{G}_2, 1)$	$K \times ((k+1)\mathbb{G}_1 + 1\mathbb{G}_T, 5)$
Sec. 5.2	$K \times (0, 2k + 3)$	$\mathbf{3}, \mathbf{0}, \mathbf{1}$	$\mathbf{6}\mathbb{G}_\mathbf{1}, \mathbf{0}$	$\mathbf{4}\mathbb{G}_\mathbf{1} + \mathbf{k}\mathbb{G}_\mathbf{2}, \mathbf{3}$
App. D	$K \times (0, 2N + 2)$	$\mathbf{3}, \mathbf{0}, \mathbf{1}$	$\mathbf{6}\mathbb{G}_\mathbf{1}, \mathbf{0}$	$\mathbf{4}\mathbb{G}_\mathbf{1} + \mathbf{2KN}\mathbb{G}_\mathbf{2}, \mathbf{3}$

Figure 1: Comparison of different ABC systems.

showings of relations (r) of attributes. All the other known constructions allow, at best, selective (s) disclosures of attributes.

In [CL11], Canard and Lescuyer use aggregate signatures to construct an ABC system. It is thus the closest to our approach. Instead of having *tags*, their signatures take *indices* as input. We follow a similar path but, we completely formalize this notion of tag/index with an EphemerId scheme. To our knowledge, aggregate signatures are the only way to deal with multiple credential issuers but still showing a unique compact credential for the proof of possession of attributes coming from different credential issuers. However, the time-complexity of a prover during a verification depends on the number k of shown attributes. We solve this issue at the cost of a larger key for the credential issuers (but still in the same order as [FHS19, San20]) and a significantly better showing cost for the prover (also better than [FHS19, San20]).

1.3 Comparison of Different ABC Systems

In Figure 1, we provide some comparisons with the most efficient ABC schemes, where the column "P" (for policy) specifies whether the scheme just allows selective disclosure of attributes (s) or relations between attributes (r). The column "T" (for traceability) indicates whether traceability is possible or not. Then, "|CI key|" gives the size of the keys (public keys of the credential issuers) required to verify the credentials, "|Show|" is the communication bandwidth during a show, while "Prover" and "Verifier" are the computational cost during a show, for the prover and the verifier respectively. Bandwidths are in number of elements $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and $\mathbb{Z}_p$. Computations are in number of exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and of pairings. Due to their negligible impact on performance, we ignore multiplications. We denote N the global number of attributes owned by a user, k the number of attributes he wants to show and K the number of credential issuers involved in the issuing of the credentials. In the first table, we focus on the particular case of proving a credential with k attributes, among N attributes issued from 1 credential issuer. Our first scheme, from Section 5.2, is already the most efficient, but this is even better for a larger K, as shown in the second table. However this is for a limited number of attributes. Our second scheme, in appendix D, has similar efficiency, but with less limitations on the attributes. Note that both schemes have a constant-size communication for the showing of any number of attributes, and the computation cost for the prover is almost constant too (as we ignore multiplications). Our two instantiations are derived from the second linearly-homomorphic signature scheme of [HPP19, Appendix C.5]. As already said, our scheme

is the first traceable attribute-based anonymous credential scheme, hence the only one in the tables.

2 Preliminaries

A reminder on the classical notations and assumptions (namely DL and DDH) used in this paper is given in appendix B. In an asymmetric bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$, or just in a simple group $\mathbb{G}$, we can define the following assumptions:

Definition 1 (Square Discrete Logarithm (SDL) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, given $y = g^x$ and $z = g^{x^2}$, it is computationally hard to recover x.

Definition 2 (Decisional Square Diffie-Hellman (DSqDH) Assumption). In a group $\mathbb{G}$ of prime order p, it states that the two following distributions are computationally indistinguishable:

$$\mathcal{D}_{\mathsf{sqdh}} = \{(g, g^x, g^{x^2}); g \xleftarrow{\$} \mathbb{G}, x \xleftarrow{\$} \mathbb{Z}_p\} \qquad \mathbb{G}_{\$}^3 = \{(g, g^x, g^y); g \xleftarrow{\$} \mathbb{G}, x, y \xleftarrow{\$} \mathbb{Z}_p\}.$$

It is worth noticing that the DSqDH Assumption implies the SDL Assumption: if one can break SDL, from g, g^x, g^{x^2}, one can compute x and thus break DSqDH. Such Square Diffie-Hellman triples will be our tags, or ephemeral public keys. For anonymity, we will use the following theorem:

Theorem 3. *The* DDH *and* DSqDH *assumptions imply the indistinguishability between the two distributions, for* $g_0, g_1 \xleftarrow{\$} \mathbb{G}$ *and* $x, y \xleftarrow{\$} \mathbb{Z}_p$

$$\mathcal{D}_0 = \{(g_0, g_0^x, g_0^{x^2}, g_1, g_1^x, g_1^{x^2})\} \qquad \mathcal{D}_1 = \{(g_0, g_0^x, g_0^{x^2}, g_1, g_1^y, g_1^{y^2})\}.$$

Proof. For this indistinguishability, one can show they are both indistinguishable from random independent 6-tuples (the distribution $\mathbb{G}_{\$}^6$):

$$\begin{aligned}
\mathcal{D}_0 &\approx \{(g_0, g_0^x, g_0^y, g_1, g_1^x, g_1^y), g_0, g_1 \xleftarrow{\$} \mathbb{G}, x, y \xleftarrow{\$} \mathbb{Z}_p\} && \text{under DSqDH}\\
&\approx \{(g_0, g_0^x, g_0^y, g_1, g_1^u, g_1^v), g_0, g_1 \xleftarrow{\$} \mathbb{G}, x, y, u, v \xleftarrow{\$} \mathbb{Z}_p\} = \mathbb{G}_{\$}^6 && \text{under DDH}\\
&\approx \{(g_0, g_0^x, g_0^{x^2}, g_1, g_1^u, g_1^{u^2}), g_0, g_1 \xleftarrow{\$} \mathbb{G}, x, u \xleftarrow{\$} \mathbb{Z}_p\} = \mathcal{D}_1 && \text{under DSqDH}
\end{aligned}$$

For unforgeability in our construction, we will use the following theorem on Square Diffie-Hellman tuples, stated and proven in [HPP19, Appendix C.5]:

Theorem 4. *Given* n *valid Square Diffie-Hellman tuples* $(g_i, a_i = g_i^{w_i}, b_i = a_i^{w_i})$*, together with* w_i*, for random* $g_i \xleftarrow{\$} \mathbb{G}^*$ *and* $w_i \xleftarrow{\$} \mathbb{Z}_p^*$*, outputting* $(\alpha_i)_{i=1,\dots,n}$ *such that* $(G = \prod g_i^{\alpha_i}, A = \prod a_i^{\alpha_i}, B = \prod b_i^{\alpha_i})$ *is a valid Square Diffie-Hellman tuple, with at least two non-zero coefficients* α_i*, is computationally hard under the* DL *assumption.*

Intuitively, from Square Diffie-Hellman tuples where the exponents are known but random (and so distinct with overwhelming probability) and the bases are also known and random, it is impossible to construct a new Square Diffie-Hellman tuple melting the exponents (with linear combinations).

3 Multi-Authority Anonymous Credentials

In this section, we define a *multi-authority* anonymous attribute-based credential scheme by adapting the model of [FHS19, San20] to the multiple credential issuers, and then, provide the associated security definitions.

3.1 Definition

Throughout the paper, we will consider the certified key setting. Indeed, we assume a Certification Authority (CA) first checks the knowledge of the secret keys before certifying public keys and then, that the keys are always checked before being used by any players in the system. Moreover, we assume that an identity id is associated (and included) to any verification key vk, which is in turn included in the secret key sk.

Our general definition of anonymous credential scheme supports multiple users $(\mathcal{U}_i)_i$ and multiple credential issuers $(\mathsf{CI}_j)_j$:

Definition 5 (Anonymous Credential). An anonymous credential system is defined by the following algorithms:

$\mathsf{Setup}(1^\kappa)$: It takes as input a security parameter and outputs the public parameters param;
$\mathsf{CIKeyGen}(\mathsf{ID})$: It generates the key pair $(\mathsf{sk}, \mathsf{vk})$ for the credential issuer with identity ID;
$\mathsf{UKeyGen}(\mathsf{id})$: It generates the key pair $(\mathsf{usk}, \mathsf{uvk})$ for the user with identity id;
$(\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}, a), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a))$: A user with identity id (associated to $(\mathsf{usk}, \mathsf{uvk})$) runs $\mathsf{CredObtain}$ to obtain a credential on the attribute a from the credential issuer ID (associated to $(\mathsf{sk}, \mathsf{vk})$) running $\mathsf{CredIssue}$. At the end of the protocol, the user receives a credential σ;
$(\mathsf{CredShow}(\mathsf{usk}, (\mathsf{vk}_j, a_j, \sigma_j)_j), \mathsf{CredVerify}((\mathsf{vk}_j, a_j)_j))$: In this two-party protocol, a user with identity id (associated to $(\mathsf{usk}, \mathsf{uvk})$) runs $\mathsf{CredShow}$ and interacts with a verifier running $\mathsf{CredVerify}$ to prove that he owns valid credentials $(\sigma_j)_j$ on $(a_j)_j$ issued respectively by credential issuers ID_j (associated to $(\mathsf{sk}_j, \mathsf{vk}_j)$). At the end of the protocol, the verifier receives 1 if the proof is correct and 0 otherwise;

3.2 Security Model

As for the definition, we follow [FHS19, San20] for the security model, with multi-show unlinkable credentials, but considering multiple credential issuers. Informally, the scheme needs to have the three properties:

- Correctness: the verifier must accept any set of credentials honestly obtained;
- Unforgeability: the verifier should not accept a set of credentials if one of them has not been legitimately obtained by this user;
- Anonymity: credentials shown multiple times by a user should be unlinkable, even for the possibly malicious credential issuers. This furthermore implies that credentials cannot be linked to their owners.

Definition 6 (Correctness). An anonymous credential scheme is said correct if, for any user id, any set of honest credential issuers HCI, and any set A of attributes:

$$\begin{aligned}
\mathsf{param} &\leftarrow \mathsf{Setup}(1^\kappa),\\
(\mathsf{usk}, \mathsf{uvk}) &\leftarrow \mathsf{UKeyGen}(\mathsf{id}),\\
(\mathsf{sk}_j, \mathsf{vk}_j) &\leftarrow \mathsf{CIKeyGen}(\mathsf{ID}_j) \text{ for } \mathsf{ID}_j \in \mathsf{HCI},\\
\sigma_j &\leftarrow (\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}_j, a_j), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}_j, a_j)) \text{ for } a_j \in A,
\end{aligned}$$

then, $1 \leftarrow (\mathsf{CredShow}(\mathsf{usk}, (\mathsf{vk}_j, a_j, \sigma_j)_j), \mathsf{CredVerify}((\mathsf{vk}_j, a_j)_j))$.

For the two security notions of unforgeability and anonymity, one can consider malicious adversaries able to corrupt some parties. We thus define the following lists: HU the list of honest user identities, CU the list of corrupted user identities, similarly we define HCI and CCI for the honest/corrupted credential issuers. For a user identity id, we define $\mathsf{Att}[\mathsf{id}]$ the list of the attributes of id and $\mathsf{Cred}[\mathsf{id}]$ the list of his individual credentials obtained from the credential

issuers. All these lists are initialized to the empty set. For both unforgeability and anonymity, the adversary has unlimited access to the oracles (in any order, but queries are assumed to be atomic):

- $\mathcal{O}\mathsf{HCI}(\mathsf{ID})$ corresponds to the creation of an honest credential issuer with identity ID. If he already exists (i.e. $\mathsf{ID} \in \mathsf{HCI} \cup \mathsf{CCI}$), it outputs $\perp$. Otherwise, it adds $\mathsf{ID} \in \mathsf{HCI}$ and runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{CIKeyGen}(\mathsf{ID})$ and returns vk;
- $\mathcal{O}\mathsf{CCI}(\mathsf{ID}, \mathsf{vk})$ corresponds to the corruption of a credential issuer with identity ID and optionally public key vk. If he does not exist yet (i.e. $\mathsf{ID} \notin \mathsf{HCI} \cup \mathsf{CCI}$), it creates a new corrupted credential issuer with public key vk by adding ID to CCI. Otherwise, if $\mathsf{ID} \in \mathsf{HCI}$, it removes ID from HCI and adds it to CCI and outputs sk;
- $\mathcal{O}\mathsf{HU}(\mathsf{id})$ corresponds to the creation of an honest user with identity id. If the user already exists (i.e. $\mathsf{id} \in \mathsf{HU} \cup \mathsf{CU}$), it outputs $\perp$. Otherwise, it creates a new user by adding $\mathsf{id} \in \mathsf{HU}$ and running $(\mathsf{usk}, \mathsf{uvk}) \leftarrow \mathsf{UKeyGen}(\mathsf{id})$. It initializes $\mathsf{Att}[\mathsf{id}] = \{\}$ and $\mathsf{Cred}[\mathsf{id}] = \{\}$ and returns uvk;
- $\mathcal{O}\mathsf{CU}(\mathsf{id}, \mathsf{uvk})$ corresponds to the corruption of a user with identity id and optionally public key uvk. If the user does not exist yet (i.e. $\mathsf{id} \notin \mathsf{HU} \cup \mathsf{CU}$), it creates a new corrupted user with public key uvk by adding id to CU. Otherwise, if $\mathsf{id} \in \mathsf{HU}$, it removes id from HU and adds it to CU and outputs usk and all the associated credentials $\mathsf{Cred}[\mathsf{id}]$;
- $\mathcal{O}\mathsf{ObtIss}(\mathsf{id}, \mathsf{ID}, a)$ corresponds to the issuing of a credential from an honest credential issuer with identity ID (associated to $(\mathsf{sk}, \mathsf{vk})$) to an honest user with identity id (associated to $(\mathsf{usk}, \mathsf{uvk})$) on the attribute a. If $\mathsf{id} \notin \mathsf{HU}$ or $\mathsf{ID} \notin \mathsf{HCI}$, it outputs $\perp$. Otherwise, it runs $\sigma \leftarrow (\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}, \mathsf{id}), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a))$ and adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$. The adversary receives the full transcript;
- $\mathcal{O}\mathsf{Obtain}(\mathsf{id}, \mathsf{ID}, a)$ corresponds to the issuing of a credential from the adversary playing the role of a malicious credential issuer with identity ID (associated to vk) to an honest user with identity id (associated to $(\mathsf{usk}, \mathsf{uvk})$) on the attribute a. If $\mathsf{id} \notin \mathsf{HU}$ or $\mathsf{ID} \notin \mathsf{CCI}$, it outputs $\perp$. Otherwise, it runs $\mathsf{CredObtain}(\mathsf{usk}, a)$ and adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$;
- $\mathcal{O}\mathsf{Issue}(\mathsf{id}, \mathsf{ID}, a)$ corresponds to the issuing of a credential from an honest credential issuer with identity ID (associated to $(\mathsf{sk}, \mathsf{vk})$) to the adversary playing the role of a malicious user with identity id (associated to uvk) on the attribute a. If $\mathsf{id} \notin \mathsf{CU}$ or $\mathsf{ID} \notin \mathsf{HCI}$, it outputs $\perp$. Otherwise, it runs $\mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a)$ and adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$;
- $\mathcal{O}\mathsf{Show}(\mathsf{id}, (\mathsf{ID}_j, a_j)_j)$ corresponds to the showing by an honest user with identity id (associated to $(\mathsf{usk}, \mathsf{uvk})$) of credentials on the set $(\mathsf{ID}_j, a_j)_j \subset \mathsf{Att}[\mathsf{id}]$ (with ID_j associated to vk_j). If $\mathsf{id} \notin \mathsf{HU}$ or $\exists j, (\mathsf{ID}_j, a_j) \notin \mathsf{Att}[\mathsf{id}]$, it outputs $\perp$. Otherwise, it runs $\mathsf{CredShow}(\mathsf{usk}, (\mathsf{vk}_j, a_j, \sigma_j)_j)$ with the adversary playing the role of a malicious verifier.

Let $\mathcal{O}$ be the list of all the above oracles: $\mathcal{O} = \{\mathcal{O}\mathsf{HCI}, \mathcal{O}\mathsf{CCI}, \mathcal{O}\mathsf{HU}, \mathcal{O}\mathsf{CU}, \mathcal{O}\mathsf{ObtIss}, \mathcal{O}\mathsf{Obtain}, \mathcal{O}\mathsf{Issue}, \mathcal{O}\mathsf{Show}\}$.

Definition 7 (Unforgeability). An anonymous credential scheme is said unforgeable if, for any polynomial time adversary $\mathcal{A}$ having access to oracles listed in $\mathcal{O}$, the advantage $\mathsf{Adv}^{\mathrm{unf}}(\mathcal{A}) = \Pr[\mathsf{Exp}^{\mathrm{unf}}_{\mathcal{A}}(1^\kappa) = 1]$ is negligible with $\mathsf{Exp}^{\mathrm{unf}}_{\mathcal{A}}(1^\kappa)$ defined in Figure 2.

Intuitively, the unforgeability notion guarantees it is impossible to make accepted a *bad* credential to a verifier. To be able to interact with the verifier, the adversary needs to corrupt at least one user and can corrupt credential issuer(s). Hence, in the security experiment in Figure 2, the adversary chooses the honest and malicious credential issuers: for the honest ones, the adversary provides their identities and the attributes in which they have authority for; while for the malicious ones, the adversary directly gives public keys and attributes. Attributes from the corrupted credential issuers can be generated by the adversary itself, using the secret keys.

Thus, the adversary wins the security game if it manages to prove its ownership of a credential, on behalf of a corrupted user $\mathsf{id} \in \mathsf{CU}$ whereas this user did not ask the attributes to the

$\mathsf{Exp}^{\mathrm{unf}}_{\mathcal{A}}(1^\kappa)$:
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$
$((\mathsf{ID}_j, a_j)_{j\in J}, (\mathsf{vk}_j, a_j)_{j\in J^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{param})$
$\forall j \in J, (\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{CIKeyGen}(\mathsf{ID}_j)$
$b \leftarrow (\mathcal{A}^{\mathcal{O}}((\mathsf{vk}_j)_{j\in J}), \mathsf{CredVerify}((\mathsf{vk}_j, a_j)_{j\in J\cup J^*}))$
If $\exists \mathsf{id} \in \mathsf{CU}$, $\forall j$, either $\mathsf{ID}_j \in \mathsf{CCI}$, or ($\mathsf{ID}_j \in \mathsf{HCI}$ and $(\mathsf{ID}_j, a_j) \in \mathsf{Att}[\mathsf{id}]$),
then return 0
Return b

$\mathsf{Exp}^{\mathrm{ano}-b}_{\mathcal{A}}(1^\kappa)$:
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$
$(\mathsf{id}_0, \mathsf{id}_1, (\mathsf{ID}_j, a_j)_{j\in J}, (\mathsf{vk}_j, a_j)_{j\in J^*}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{param})$
If for some $\mathsf{ID}_j \in \mathsf{CCI} \cup \mathsf{HCI}$, $(\mathsf{ID}_j, a_j) \notin \mathsf{Att}[\mathsf{id}_0] \cap \mathsf{Att}[\mathsf{id}_1]$, then return 0
$\forall j \in J$,
$(\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{CIKeyGen}(\mathsf{ID}_j)$
$\sigma_{0,j} \leftarrow (\mathsf{CredObtain}(\mathsf{usk}_0, \mathsf{vk}_j, a_j), \mathsf{CredIssue}(\mathsf{uvk}_0, \mathsf{sk}_j, a_j))$
$\sigma_{1,j} \leftarrow (\mathsf{CredObtain}(\mathsf{usk}_1, \mathsf{vk}_j, a_j), \mathsf{CredIssue}(\mathsf{uvk}_1, \mathsf{sk}_j, a_j))$
$(\mathsf{CredShow}(\mathsf{usk}_b, (a_j, \sigma_{b,j})_j), \mathcal{A}^{\mathcal{O}}((\mathsf{vk}_j, \sigma_{0,j}, \sigma_{1,j})_{j\in J}, \mathsf{uvk}_0, \mathsf{uvk}_1))$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}}()$
If $\mathsf{id}_0 \in \mathsf{CU}$ or $\mathsf{id}_1 \in \mathsf{CU}$, then return 0
Return b^*

In both experiments, the adversary $\mathcal{A}^{\mathcal{O}}$ has unlimited access to the oracles $\mathcal{O} = \{\mathcal{O}\mathsf{HCI}, \mathcal{O}\mathsf{CCI}, \mathcal{O}\mathsf{HU}, \mathcal{O}\mathsf{CU}, \mathcal{O}\mathsf{ObtIss}, \mathcal{O}\mathsf{Obtain}, \mathcal{O}\mathsf{Issue}, \mathcal{O}\mathsf{Show}\}$. It is also supposed to maintain an internal state all along the games.

Figure 2: Unforgeability and Anonymity Experiments

honest credential issuers. In other words, if among the corrupted users there is one ($\mathsf{id} \in \mathsf{CU}$) having credentials on all the proposed attributes for a forgery issued by either:

- corrupted credential issuers,
- honest credential issuers (in that case, for ID_j honest, (ID_j, a_j) is in $\mathsf{Att}[\mathsf{id}]$),

then, the forgery is a trivial one and excluded by definition. This is not a legitimate attack.

Definition 8 (Anonymity). An anonymous credential scheme is said anonymous if, for any polynomial time adversary $\mathcal{A}$ having access to oracles listed in $\mathcal{O}$, the advantage $\mathsf{Adv}^{\mathrm{ano}}(\mathcal{A}) = |\Pr[\mathsf{Exp}^{\mathrm{ano}-1}_{\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Exp}^{\mathrm{ano}-0}_{\mathcal{A}}(1^\kappa) = 1]|$ is negligible with $\mathsf{Exp}^{\mathrm{ano}-b}_{\mathcal{A}}(1^\kappa)$ defined in Figure 2.

The anonymity is defined by an unlinkability notion: the adversary tries to distinguish if two showings come from the same user or not. More precisely, the adversary wins the security game if it can distinguish showings from (honest) users id_0 and id_1 of its choice, on the same set of attributes $\{(\mathsf{ID}_j, a_j)\}_j$, even after having seen/verified credentials from the two identities, as it has access to the oracle $\mathcal{O}\mathsf{Show}$ to interact with them. Note that we do not hide the attributes nor the issuers during the showings: uvk_0 and uvk_1 are given to the adversary.

As for the unforgeability, the adversary is authorized to corrupt as many credential issuers as it wants (possibly all of them) and at any time thanks to the oracle $\mathcal{O}\mathsf{CCI}$. For the honest ones, the adversary provides their identities and the attributes in which they have authority for; while for the malicious ones, the adversary directly gives their public keys and their attributes.

Contrarily to [San20], unless the attributes contain explicit ordering (as it will be the case with our first construction), we are dealing with unlinkability as soon as the sets of attributes are the same for the two players (with the second construction).

4 Anonymous Credentials from New Primitives

Before we present our credential scheme, let us formally define the way users will be identified.

4.1 Anonymous Ephemeral Identities

In attribute-based authentication, a credential usually consists of a signature on some attributes together with the public key of a user. The identity of the user is then ensured by the correctness of its key provided by a Certification Authority (CA). Hence, one can represent the identity of a user in an anonymous credential scheme by a pair $(\tilde{\tau}, \tau)$, where $\tilde{\tau}$ is a secret tag and τ the associated public tag, that requires to be randomizable for anonymity.

Formally, one defines this tag pair as an ephemeral identity provided by an $\mathsf{EphemerId}$ scheme:

Definition 9 ($\mathsf{EphemerId}$). An $\mathsf{EphemerId}$ scheme consists of the algorithms:

- $\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, it outputs the global parameter param, which includes the tag space $\mathcal{T}$;
- $\mathsf{GenTag}(\mathsf{param})$: Given a public parameter param, it outputs a secret tag $\tilde{\tau}$ and an associated public tag τ;
- $\mathsf{RandTag}(\tau)$: Given a public tag τ as input, it outputs a new tag τ' and the randomization link $\rho_{\tau\to\tau'}$ between τ and τ';
- $\mathsf{DerivWitness}(\tilde{\tau}, \rho_{\tau\to\tau'})$: Given a witness $\tilde{\tau}$ (associated to the tag τ) and a link between the tags τ and τ' as input, it outputs a witness $\tilde{\tau}'$ for τ';
- $(\mathsf{ProveVTag}(\tilde{\tau}), \mathsf{VerifVTag}(\tau))$: This (possibly interactive) protocol corresponds to the verification of the tag τ. At the end of the protocol, the verifier outputs 1 if it accepts τ as a valid tag and 0 otherwise;
- $(\mathsf{ProveKTag}(s, \tilde{\tau}), \mathsf{VerifKTag}(s, \tau))$: This (possibly interactive) protocol corresponds to a fresh proof of knowledge of $\tilde{\tau}$ using the state s. At the end of the protocol, the verifier outputs 1 if it accepts the proof and 0 otherwise.

With the above algorithms, one can define the sets:

$$\mathcal{L}_{\tilde{\tau}} = \{\tau \in \mathcal{T}, \tau \text{ public tag associated to } \tilde{\tau}\} \qquad \mathcal{L} = \cup_{x\in\mathbb{Z}_p^*}\mathcal{L}_x$$

and, as the set of all the $\mathcal{L}_{\tilde{\tau}}$ is a partition of $\mathcal{L}$, an equivalence relation $\sim$ between tags: $\tau \sim \tau' \Leftrightarrow \exists\tilde{\tau}, (\tau, \tau' \in \mathcal{L}_{\tilde{\tau}})$. Hence, each authorized user will be associated to a secret tag $\tilde{\tau}$, and represented by the class generated by the public tag τ. One may have to prove the actual membership $\tau \in \mathcal{L}$ to prove the user is authorized. On the other hand, one may also have to prove the knowledge of the witness $\tilde{\tau}$, in a zero-knowledge way, for authentication.

The latter proof of knowledge can be performed, using the (interactive) protocol $(\mathsf{ProveKTag}(\tilde{\tau}), \mathsf{VerifKTag}(\tau))$. Interactive protocol or signature of knowledge on a fresh message will be useful for the freshness in the authentication process, and to avoid replay attacks. The former proof of validity can also be proven using an (interactive) protocol $(\mathsf{ProveVTag}(\tilde{\tau}), \mathsf{VerifVTag}(\tau))$. However this verification can also be non-interactive or even public, without needing any private witness. The only requirement is that this proof or verification of membership should not reveal the private witness involved in the proof of knowledge, whose soundness will guarantee the authentication of the user.

Security. The security notions are the usual ones for zero-knowledge proofs for the protocols $(\mathsf{ProveKTag}(\tilde{\tau}), \mathsf{VerifKTag}(\tau))$ and $(\mathsf{ProveVTag}(\tilde{\tau}), \mathsf{VerifVTag}(\tau))$:

- Soundness: the verification process for the validity of the tag should not accept an invalid tag (not in the language);
- Knowledge-Soundness: if the verification process for the proof of knowledge of the witness accepts with good probability, a simulator can extract it;
- Zero-knowledge: the proof of validity and the proof of knowledge should not reveal any information about the witness.

When the two protocols output 1, the witness-word pair is said to be valid.

Correctness. For an honestly generated pair $(\tilde{\tau}, \tau) \leftarrow \mathsf{GenTag}(\mathsf{param})$, the witness-word pair must be valid (i.e. both protocols $(\mathsf{ProveVTag}(\tilde{\tau}), \mathsf{VerifVTag}(\tau))$ and $(\mathsf{ProveKTag}(s, \tilde{\tau}), \mathsf{VerifKTag}(s, \tau))$ must output 1).

From an honestly generated witness-word pair $(\tilde{\tau}, \tau) \leftarrow \mathsf{GenTag}(\mathsf{param})$, if $(\tau', \rho) \leftarrow \mathsf{RandTag}(\tau)$ and $\tilde{\tau}' \leftarrow \mathsf{DerivWitness}(\tilde{\tau}, \rho)$ then $(\tilde{\tau}', \tau')$ must also be a valid witness-word pair.

Unlinkability. The algorithm $\mathsf{RandTag}$ must randomize the tag τ within the equivalence class in an unlinkable way: for any pair $((\tilde{\tau}_1, \tau_1), (\tilde{\tau}_2, \tau_2))$ issued from GenTag, the two distributions $\{(\tau_1, \tau_2, \tau_1', \tau_2')\}$ and $\{(\tau_1, \tau_2, \tau_2', \tau_1')\}$, where $\tau_1' \leftarrow \mathsf{RandTag}(\tau_1)$ and $\tau_2' \leftarrow \mathsf{RandTag}(\tau_2)$, must be (computationally) indistinguishable.

4.2 Tag-based Signatures

For a pair $(\tilde{\tau}, \tau)$, where $\tilde{\tau}$ is a secret tag and τ the associated public tag, one can define a new primitive called *tag-based signature*:

Definition 10 (Tag-Based Signature).

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, it outputs the global parameter param, which includes the message space $\mathcal{M}$ and the tag space $\mathcal{T}$;

$\mathsf{Keygen}(\mathsf{param})$: Given a public parameter param, it outputs a key pair $(\mathsf{sk}, \mathsf{vk})$;

$\mathsf{GenTag}(\mathsf{param})$: Given a public parameter param, it generates a witness-word pair $(\tilde{\tau}, \tau)$;

$\mathsf{Sign}(\mathsf{sk}, \tau, m)$: Given a signing key sk, a tag τ, and a message m, it outputs the signature σ under the tag τ;

$\mathsf{VerifSign}(\mathsf{vk}, \tau, m, \sigma)$: Given a verification key vk, a tag τ, a message m and a signature σ, it outputs 1 if σ is valid relative to vk and τ, and 0 otherwise.

The security notion would expect no adversary able to forge, for any honest pair $(\mathsf{sk}, \mathsf{vk})$, a new signature for a pair (τ, m), for a valid tag τ, if the signature has not been generated using sk and the tag τ on the message m.

Two classical cases are: $(\tilde{\tau} = \mathsf{sk}, \tau = \mathsf{vk})$, which corresponds to a classical signature of m; $\tilde{\tau} = \tau$, with no secret witness, this is just a classical signature of (τ, m). In fact, more subtle situations can be handled, as shown below.

Signatures with Randomizable Tags. When randomizing τ into τ', one must be able to keep track of the change to update $\tilde{\tau}$ to $\tilde{\tau}'$ and the signatures. Formally, we will require to have the algorithm:

$\mathsf{DerivSign}(\mathsf{vk}, \tau, m, \sigma, \rho_{\tau \to \tau'})$: Given a valid signature σ on tag τ and message m, and $\rho_{\tau \to \tau'}$ the randomization link between τ and another tag τ', it outputs a new signature σ' on the message m and the new tag τ'. Both signatures are under the same key vk.

For compatibility with the tag and correctness of the signature scheme, we require that for all honestly generated keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(\mathsf{param})$, all tags $(\tilde{\tau}, \tau) \leftarrow \mathsf{GenTag}(\mathsf{param})$, and all messages m, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m)$, $(\tau', \rho) \leftarrow \mathsf{RandTag}(\tau)$ and $\sigma' \leftarrow \mathsf{DerivSign}(\mathsf{vk}, \tau, m, \sigma, \rho)$, then $\mathsf{VerifSign}(\mathsf{vk}, \tau', m, \sigma')$ should output 1.

For privacy reasons, in case of probabilistic signatures, it will not be enough to just randomize the tag, but the random coins of the signing algorithm too:

$\mathsf{RandSign}(\mathsf{vk}, \tau, m, \sigma)$: Given a valid signature σ on tag τ and message m, it outputs a new signature σ' on the same message m and tag τ.

Correctness extends the above one, where the algorithm $\mathsf{VerifSign}(\mathsf{vk}, \tau', m, \sigma'')$ should output 1 with $\sigma'' \leftarrow \mathsf{RandSign}(\mathsf{vk}, \tau', m, \sigma')$. One additionally expects unlinkability: the distributions $\mathcal{D}_0$

and $\mathcal{D}_1$ are (computationally) indistinguishable, for any vk and m (possibly chosen by the adversary), where for $i = 0, 1$, $(\tilde{\tau}_i, \tau_i) \leftarrow \mathsf{GenTag}(1^\kappa)$, $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau_i, m)$, $(\tau'_i, \rho_i) \leftarrow \mathsf{RandTag}(\tau_i)$, $\sigma'_i \leftarrow \mathsf{DerivSign}(\mathsf{vk}, \tau_i, m, \sigma_i, \rho_i)$ and $\sigma''_i \leftarrow \mathsf{RandSign}(\mathsf{vk}, \tau'_i, m, \sigma'_i)$, and for $b = 0, 1$, we set $\mathcal{D}_b = \{(m, \mathsf{vk}, \tau_0, \sigma_0, \tau'_b, \sigma''_b, \tau_1, \sigma_1, \tau'_{1-b}, \sigma''_{1-b})\}$. We stress that this indistinguishability should also hold with respect to the signer, who knows the signing key, after randomization of the signature (and not just of the tag) in case of probabilistic signature.

Signatures with Randomizable Tags on Vectors. We will extend the above algorithms to vectors of keys $\vec{\mathsf{sk}}$, $\vec{\mathsf{pk}}$, messages $\vec{m}$, and signatures $\vec{\sigma}$, of the same length, but for a common tag τ, by applying the algorithms on the component-wise elements:

- $\mathsf{Sign}(\vec{\mathsf{sk}}, \tau, \vec{m})$ outputs $\vec{\sigma}$ where each component $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \tau, m_i)$, for the common tag τ;
- $\mathsf{VerifSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma})$ outputs the conjunction of the Boolean output by all the verifications $\mathsf{VerifSign}(\mathsf{vk}_i, \tau, m_i, \sigma_i)$ on each component, under the common tag τ (whether all the verifications accept or not);
- $\mathsf{DerivSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma}, \rho_{\tau\to\tau'})$ outputs $\vec{\sigma'}$ where each component is derived as $\sigma'_i \leftarrow \mathsf{DerivSign}(\mathsf{vk}_i, \tau, m_i, \sigma_i, \rho_{\tau\to\tau'})$;
- $\mathsf{RandSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma})$ outputs $\vec{\sigma'}$ where each component is randomized as $\sigma'_i \leftarrow \mathsf{RandSign}(\mathsf{vk}_i, \tau, m_i, \sigma_i)$, for the common tag τ.

This is a preliminary step towards aggregate signatures that will allow a compact representation of $\vec{\sigma}$, independent of the length of the vector, when the tag τ is the same for all the components. This will be the core of our compact anonymous credentials. In both cases, with vectors of any length or of length 1, some properties will be required.

Correctness. From any valid tag-pair $(\tilde{\tau}, \tau)$ and honestly generated pairs of keys $(\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{Keygen}(\mathsf{param})$, if $\sigma_j = \mathsf{Sign}(\mathsf{sk}_j, \tau, m_j)$ are valid signatures on message $m_j \in \mathcal{M}$, for $j = 1, \cdots, \ell$, if $(\tau', \rho) \leftarrow \mathsf{RandTag}(\tau)$:

- either after the derivation for the tag, $\vec{\sigma'} \leftarrow \mathsf{DerivSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma}, \rho)$, and the randomization of the signature, $\vec{\sigma''} \leftarrow \mathsf{RandSign}(\vec{\mathsf{vk}}, \tau', \vec{m}, \vec{\sigma'})$
- or after the randomization of the signature $\vec{\sigma'} \leftarrow \mathsf{RandSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma})$, and the derivation for the tag, $\vec{\sigma''} \leftarrow \mathsf{DerivSign}(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma'}, \rho)$

then the verification $\mathsf{VerifSign}(\vec{\mathsf{vk}}, \tau', \vec{m}, \vec{\sigma''})$ should output 1.

Unforgeability. In the Chosen-Message Unforgeability security game, the adversary has unlimited access to the following oracles, with lists KList and TList initially empty:

- $\mathcal{O}\mathsf{GenTag}()$ outputs the tag τ and keeps track of the associated witness $\tilde{\tau}$, with $(\tilde{\tau}, \tau)$ appended to TList;
- $\mathcal{O}\mathsf{Keygen}()$ outputs the verification key vk and keeps track of the associated signing key sk, with $(\mathsf{sk}, \mathsf{vk})$ appended to KList;
- $\mathcal{O}\mathsf{Sign}(\tau, \mathsf{vk}, m)$, for $(\tilde{\tau}, \tau) \in \mathsf{TList}$ and $(\mathsf{sk}, \mathsf{vk}) \in \mathsf{KList}$, outputs $\mathsf{Sign}(\mathsf{sk}, \tau, m)$.

It should not be possible to generate a signature that falls outside the range of $\mathsf{DerivSign}$ or $\mathsf{RandSign}$:

Definition 11 (Unforgeability for RT-Sign). An RT-Sign scheme is said unforgeable if, for any adversary $\mathcal{A}$ that, given signatures σ_i for tuples $(\tau_i, \mathsf{vk}_i, m_i)$ of its choice but for τ_i and vk_i issued from the GenTag and Keygen algorithms respectively (for Chosen-Message Attacks), outputs a tuple $(\vec{\mathsf{vk}}, \tau, \vec{m}, \vec{\sigma})$ where both τ is a valid tag and $\vec{\sigma}$ is a valid signature w.r.t. $(\vec{\mathsf{vk}}, \tau, \vec{m})$, there exists a subset J of the signing queries with a common tag $\tau' \in \{\tau_i\}_i$ such that $\tau \sim \tau'$, $\forall j \in J, \tau_j = \tau'$, $\vec{\mathsf{vk}} = (\mathsf{vk}_j)_{j\in J}$, and $\vec{m} = (m_j)_{j\in J}$, with overwhelming probability.

Since there are multiple secrets, we can consider corruptions of some of them:

- $\mathcal{O}\mathsf{CorruptTag}(\tau)$, for $(\tilde{\tau}, \tau) \in \mathsf{TList}$, outputs $\tilde{\tau}$;
- $\mathcal{O}\mathsf{Corrupt}(\mathsf{vk})$, for $(\mathsf{sk}, \mathsf{vk}) \in \mathsf{KList}$, outputs sk.

The forgery should not involve a corrupted key (but corrupted tags are allowed). Note again that all the tags are valid (either issued from GenTag or verified). In the unforgeability security notion, some limitations might be applied to the signing queries: one-time queries (for a given tag-key pair) or a bounded number of queries.

Unlinkability. Randomizability of both the tag and the signature are expected to provide anonymity, with some unlinkability property:

Definition 12 (Unlinkability for RT-Sign). An RT-Sign scheme is said unlinkable if, for any $\vec{\mathsf{vk}}$ and $\vec{m}$, no adversary $\mathcal{A}$ can distinguish the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where for $i = 0, 1$, we have $(\tilde{\tau}_i, \tau_i) \leftarrow \mathsf{GenTag}(1^\kappa)$, $(\tau'_i, \rho_i) \leftarrow \mathsf{RandTag}(\tau_i)$, $\vec{\sigma_i}$ is any valid signature of $\vec{m}$ under τ_i and $\vec{\mathsf{vk}}$, and eventually $\vec{\sigma'_i} \leftarrow \mathsf{DerivSign}(\vec{\mathsf{vk}}, \tau_i, \vec{m}, \vec{\sigma_i}, \rho_i)$ and $\vec{\sigma''_i} \leftarrow \mathsf{RandSign}(\vec{\mathsf{vk}}, \tau'_i, \vec{m}, \vec{\sigma'_i})$, and for $b = 0, 1$ we set $\mathcal{D}_b = \{(\vec{m}, \vec{\mathsf{vk}}, \tau_0, \vec{\sigma_0}, \tau'_b, \vec{\sigma''_B}, \tau_1, \vec{\sigma_1}, \tau'_{1-b}, \vec{\sigma''_{1-b}})\}$.

We stress that this indistinguishability should also hold with respect to the signer, who might have generated the initial signatures $\vec{\sigma_0}$ and $\vec{\sigma_1}$, but then after randomization of the signature (and not just of the tag, with derivation) in case of probabilistic signature.

4.3 Anonymous Credential from EphemerId and RT-Sign

We first explain how EphemerId and RT-Sign can lead to anonymous credentials, in a black-box way. Our concrete constructions will thereafter exploit aggregate RT-Sign, to achieve compactness, independently of the number of credentials and authorities (credential issuers).

Let $\mathcal{E}$ be an EphemerId scheme and S^{rt} an RT-Sign scheme, one can construct an anonymous attribute-based credential scheme. The user's keys will be tag pairs and the credentials will be RT-Sign signatures on both the tags and the attributes. Since the tag is randomizable, the user can anonymously show any set of credentials: multiple showings will use unlinkable tags.

Furthermore, as the signature scheme tolerates corruptions of users and signers, we will be able to consider corruptions of users and credential issuers, and even possible collusions:

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, it runs $\mathsf{S}^{\mathrm{rt}}.\mathsf{Setup}$ and outputs the public parameters param which includes all the parameters;

$\mathsf{CIKeyGen}(\mathsf{ID})$: Credential issuer CI with identity ID, runs $\mathsf{S}^{\mathrm{rt}}.\mathsf{Keygen}(\mathsf{param})$ to obtain his key pair $(\mathsf{sk}, \mathsf{vk})$;

$\mathsf{UKeyGen}(\mathsf{id})$: User $\mathcal{U}$ with identity id, runs $\mathcal{E}.\mathsf{GenTag}(\mathsf{param})$ to obtain his key pair $(\mathsf{usk}, \mathsf{uvk})$;

$(\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}, a), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a))$: User $\mathcal{U}$ with identity id and key-pair $(\mathsf{usk}, \mathsf{uvk})$ asks the credential issuer CI for a credential on attribute a: $\sigma = \mathsf{S}^{\mathrm{rt}}.\mathsf{Sign}(\mathsf{sk}, \mathsf{uvk}, a)$, which can be checked by the user;

$(\mathsf{CredShow}(\mathsf{usk}, (\mathsf{vk}_j, a_j, \sigma_j)_j), \mathsf{CredVerify}((\mathsf{vk}_j, a_j)_j))$: User $\mathcal{U}$ randomizes his public key $(\mathsf{uvk}', \rho) = \mathcal{E}.\mathsf{RandTag}(\mathsf{uvk})$. Then, it adapts the secret key $\mathsf{usk}' = \mathcal{E}.\mathsf{DerivWitness}(\mathsf{usk}, \rho)$, thanks to ρ, as well as the signatures $(\sigma'_j)_j = \mathsf{S}^{\mathrm{rt}}.\mathsf{DerivSign}((\mathsf{vk}_j)_j, \mathsf{uvk}, (a_j, \sigma_j)_j, \rho)$, using the above algorithm on vectors. It then randomizes them: $(\sigma''_j)_j = \mathsf{S}^{\mathrm{rt}}.\mathsf{RandSign}((\mathsf{vk}_j)_j, \mathsf{uvk}', (a_j, \sigma'_j)_j)$. It eventually sends the anonymous credentials $((\mathsf{vk}_j, a_j, \sigma''_j)_j, \mathsf{uvk}')$ to the verifier $\mathcal{V}$. The verifier first checks the freshness of the credentials with a proof of ownership of uvk'. This is performed using the interactive protocol $(\mathcal{E}.\mathsf{ProveKTag}(\mathsf{usk}'), \mathcal{E}.\mathsf{VerifKTag}(\mathsf{uvk}'))$. It then verifies the validity of the credentials with $\mathsf{S}^{\mathrm{rt}}.\mathsf{VerifSign}((\mathsf{vk}_j)_j, \mathsf{uvk}', (a_j, \sigma''_j)_j)$.

We stress that for all the above notations $(u_j)_j$, we use the algorithms on vectors, that apply the initial algorithms component-wise. This does not lead to compact credentials, as they are vectors $(\sigma_j)_j$, but we only consider security in this section. Efficiency will be dealt in the next section, with aggregate signatures.

If one considers corruptions, when one corrupts a user or a credential issuer, the corresponding secret key is provided.

With secure EphemerId and RT-Sign schemes, the above construction is an anonymous attribute-based credential scheme, with security results stated below and proven in appendix C.

Theorem 13. *Assuming EphemerId achieves knowledge-soundness and RT-Sign is unforgeable, the generic construction is an unforgeable attribute-based credential scheme, in the certified key model.*

Theorem 14. *Assuming EphemerId is zero-knowledge and RT-Sign is unlinkable, the generic construction is an anonymous attribute-based credential scheme, in the certified key model.*

5 Constructions

Our concrete constructions will provide compact credentials. To this aim, we need more compact signatures on vectors than vectors of signatures. We thus recall the notion of aggregate signatures, and define the aggregate signatures with randomizable tags.

5.1 Aggregate Signatures with Randomizable Tags

Aggregate Signatures. Boneh et al. [BGLS03] remarked it was possible to aggregate the BLS signature [BLS01], we will follow this path, but for tag-based signatures, with possible aggregation only between signatures with the same tag, in a similar way as the indexed aggregated signatures [CL11]. We will even consider aggregation of public keys, which can either still be a simple concatenation or a more evolved combination as in [BDN18]. Hence, an aggregate (tag-based) signature scheme (Aggr-Sign) is a signature scheme with the algorithms:

$\mathsf{AggrKey}((\mathsf{vk}_j)_{j=1}^{\ell})$: Given ℓ verification keys vk_j, it outputs an aggregated verification key avk;

$\mathsf{AggrSign}(\tau, (\mathsf{vk}_j, m_j, \sigma_j)_{j=1}^{\ell})$: Given ℓ signed messages m_j in σ_j under vk_j and the same tag τ, it outputs a signature σ on the message vector $\vec{m} = (m_j)_{j=1}^{\ell}$ under the tag τ and aggregated verification key avk.

One can note that avk can be $\vec{\mathsf{vk}}$ or a more compact encoding. Similarly, the aggregate signature σ from $\vec{\sigma} = (\sigma_j)_j$ can remain this concatenation or a more compact representation.

While we will still focus on signing algorithm of a single message with a single key, we have to consider verification algorithm on vectors of keys and message, but on a compact signature.

Correctness of an aggregate (tag-based) signature scheme requires that for any valid tag-pair $(\tilde{\tau}, \tau)$ and honestly generated keys $(\mathsf{sk}_j, \mathsf{vk}_j) \leftarrow \mathsf{Keygen}(\mathsf{param})$, if $\sigma_j = \mathsf{Sign}(\mathsf{sk}_j, \tau, m_j)$ are valid signatures for $j = 1, \cdots, \ell$, then for both key $\mathsf{avk} \leftarrow \mathsf{AggrKey}((\mathsf{vk}_j)_{j=1}^{\ell})$ and signature $\sigma = \mathsf{AggrSign}(\tau, (\mathsf{vk}_j, m_j, \sigma_j)_{j=1}^{\ell})$, the verification $\mathsf{VerifSign}(\mathsf{avk}, \tau, (m_j)_{j=1}^{\ell}, \sigma)$ should output 1.

Aggregate Signatures with Randomizable Tags. We can now provide the formal definition of an aggregate signature scheme with randomizable tags, where some algorithms exploit compatibility between the EphemerId scheme and the signature scheme:

Definition 15 (Aggregate Signatures with randomizable tags (ART-Sign)). An ART-Sign scheme, associated to an EphemerId scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{GenTag}, \mathsf{RandTag}, \mathsf{DerivWitness}, (\mathsf{ProveVTag}, \mathsf{VerifVTag}))$ consists of the algorithms (Setup, Keygen, Sign, AggrKey, AggrSign, DerivSign, RandSign, VerifSign):

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, it runs $\mathcal{E}.\mathsf{Setup}$ and outputs the global parameter param, which includes $\mathcal{E}.\mathsf{param}$ with the tag space $\mathcal{T}$, and extends it with the message space $\mathcal{M}$;

$\mathsf{Keygen}(\mathsf{param})$: Given a public parameter param, it outputs a key-pair $(\mathsf{sk},\mathsf{vk})$;

$\mathsf{Sign}(\mathsf{sk}, \tau, m)$: Given a signing key, a valid tag τ, and a message $m \in \mathcal{M}$, it outputs the signature σ;

$\mathsf{AggrKey}((\mathsf{vk}_j)_{j=1}^{\ell})$: Given ℓ verification keys vk_j, it outputs an aggregated verification key avk;

$\mathsf{AggrSign}(\tau, (\mathsf{vk}_j, m_j, \sigma_j)_{j=1}^{\ell})$: Given ℓ signed messages m_j in σ_j under vk_j and the same valid tag τ, it outputs a signature σ on the vector $\vec{m} = (m_j)_{j=1}^{\ell}$ under the tag τ and aggregated verification key avk;

$\mathsf{VerifSign}(\mathsf{avk}, \tau, \vec{m}, \sigma)$: Given a verification key avk, a valid tag τ, a vector $\vec{m}$ and a signature σ, it outputs 1 if σ is valid relative to avk and τ, and 0 otherwise;

$\mathsf{DerivSign}(\mathsf{avk}, \tau, \vec{m}, \sigma, \rho_{\tau\to\tau'})$: Given a signature σ on a vector $\vec{m}$ under a valid tag τ and aggregated verification key avk, and the randomization link $\rho_{\tau\to\tau'}$ between τ and another tag τ', it outputs a signature σ' on the vector $\vec{m}$ under the new tag τ' and the same key avk;

$\mathsf{RandSign}(\mathsf{avk}, \tau, \vec{m}, \sigma)$: Given a signature σ on a vector $\vec{m}$ under a valid tag τ and aggregated verification key avk, it outputs a new signature σ' on the vector $\vec{m}$ and the same tag τ.

We stress that all the tags must be valid: their verification must be performed before the verification of the signatures.

Note that using algorithms from $\mathcal{E}$, tags are randomizable at any time, and signatures adapted and randomized, even after an aggregation: avk and $\vec{m}$ can either be single key and message or aggregations of keys and messages. Note that only protocol $(\mathsf{ProveVTag}, \mathsf{VerifVTag})$ from $\mathcal{E}$ is involved in the ART-Sign scheme, as one just needs to check the validity of the tag, not the ownership. The latter will be useful in anonymous credentials with fresh proof of ownership.

Correctness, unforgeability, and unlinkability are the same as above, where the compact aggregate signature σ replaces the vector $\vec{\sigma}$.

5.2 Constructions

We can now instantiate the different primitives. More precisely, we provide two constructions of a multi-authority anonymous credential scheme each one based on a construction of an ART-Sign scheme: a one-time version and a bounded version. In the first construction, we consider attributes where the index i determines the attribute type (age, city, diploma) and the exact value is encoded in $a_i \in \mathbb{Z}_p^*$ (possibly $\mathcal{H}(m) \in \mathbb{Z}_p^*$ if the value is a large bitstring), or 0 when empty. The second construction (see appendix D) does not require any such ordering on the attributes. Arbitrary bit strings are supported. However, the construction of the EphemerId scheme is in common.

SqDH-based EphemerId Scheme. With tags in $\mathcal{T} = \mathbb{G}_1^3$, in an asymmetric bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$, and $\tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2})$ a Square Diffie-Hellman tuple, one can define the SqDH EphemerId scheme:

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$ be an asymmetric bilinear setting, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The set of (valid and invalid) tags is $\mathcal{T} = \mathbb{G}_1^3$. We then define $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e; \mathcal{T})$;

$\mathsf{GenTag}(\mathsf{param})$: Given a public parameter param, it randomly chooses a generator $h \xleftarrow{\$} \mathbb{G}_1^*$ and outputs $\tilde{\tau} \xleftarrow{\$} \mathbb{Z}_p^*$ and $\tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$.

$\mathsf{RandTag}(\tau)$: Given a tag τ as input, it chooses $\rho_{\tau\to\tau'} \xleftarrow{\$} \mathbb{Z}_p$ and constructs $\tau' = \tau^{\rho_{\tau\to\tau'}}$ the derived tag. It outputs $(\tau', \rho_{\tau\to\tau'})$.

$\mathsf{DerivWitness}(\tilde{\tau}, \rho_{\tau\to\tau'})$: The derived witness remains unchanged: $\tilde{\tau}' = \tilde{\tau}$.

$\mathsf{ProveVTag}(\tilde{\tau}), \mathsf{VerifVTag}(\tau)$: The prover constructs the proof $\pi = \mathsf{proof}(\tilde{\tau} : \tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}))$ (see appendix F.2 for a non-interactive proof using the Groth-Sahai [GS08] framework). The verifier outputs 1 if it accepts the proof and 0 otherwise.

Valid tags are Square Diffie-Hellman pairs in $\mathbb{G}_1$:

$$\mathcal{L} = \{(h, h^x, h^{x^2}), h \in \mathbb{G}_1^*, x \in \mathbb{Z}_p^*\} = \cup_{x\in\mathbb{Z}_p^*}\mathcal{L}_x \qquad \mathcal{L}_x = \{(h, h^x, h^{x^2}), h \in \mathbb{G}_1^*\}$$

The randomization does not affect the exponents, hence there are $p-1$ different equivalence classes $\mathcal{L}_x$, for all the non-zero exponents $x \in \mathbb{Z}_p^*$, and correctness is clearly satisfied within equivalence classes. The validity check (see appendix F.2) is sound as the Groth-Sahai commitment is in the perfectly binding setting. Such tags also admit an interactive Schnorr-like zero-knowledge proof of knowledge of the exponent $\tilde{\tau}$ for $(\mathsf{ProveKTag}(\tilde{\tau}), \mathsf{VerifKTag}(\tau))$ which also provides extractability (knowledge-soundness). With the Fiat-Shamir heuristic and the random oracle, this proof of knowledge can be transformed into a non-interactive one, also called a signature of knowledge. Under the DSqDH and DL assumptions, given the tag τ, it is hard to recover the exponent $\tilde{\tau} = x$. The tags, after randomization, are uniformly distributed in the equivalence class, and under the DSqDH-assumption, each class is indistinguishable from $\mathbb{G}_1^3$, and thus one has unlinkability: see Theorem 3.

One-Time SqDH-based ART-Sign Scheme. The above Ephemerld scheme can be extended into an ART-Sign scheme where implicit vector messages are signed. As the aggregation can be made on signatures of messages under the same tag but from various signers, the description is given for multiple and independent signers, each indexed by j, and any signed message by the j-signer for coordinate i is indexed by (j, i).

We stress that this *one-time* scheme needs to be state-full as there is the limitation for a signer j not to sign more than one message with index (j, i) for a given tag: a signer must use two different indices to sign two messages for one tag. This is due to the linearly-homomorphic signature scheme: each coordinate a is signed, as a pair (g, g^a), in a subspace of dimension 2. The linearity limits to Diffie-Hellman pairs with constant ratio a. But with two independent 2-dimension vectors, one can generate the full subspace $\mathbb{G}_1 \times \mathbb{G}_1$.

Our construction of aggregate signature with randomizable tags is based on the second linearly-homomorphic signature scheme of [HPP19, Appendix C.5]:

$\mathsf{Setup}(1^\kappa)$: It extends the above setup with the set of messages $\mathcal{M} = \mathbb{Z}_p$;

$\mathsf{Keygen}(\mathsf{param})$: Given the public parameters param, it outputs the signing and verification keys

$$\mathsf{sk}_{j,i} = (\ \mathsf{SK}_j = [\ t_j,\ u_j,\ v_j\ ],\ \mathsf{SK}'_{j,i} = [\ r_{j,i},\ s_{j,i}\ ]\ ) \xleftarrow{\$} \mathbb{Z}_p^5,$$
$$\mathsf{vk}_{j,i} = (\ \mathsf{VK}_j = [\ \mathfrak{g}^{t_j}, \mathfrak{g}^{u_j}, \mathfrak{g}^{v_j}\ ],\ \mathsf{VK}'_{j,i} = [\ \mathfrak{g}^{r_{j,i}}, \mathfrak{g}^{s_{j,i}}\ ]\ ) \in \mathbb{G}_2^5.$$

Note that one could dynamically add new $\mathsf{SK}'_{j,i}$ and $\mathsf{VK}'_{j,i}$ to sign implicit vector messages: $\mathsf{sk}_j = \mathsf{SK}_j \cup [\mathsf{SK}'_{j,i}]_i$, $\mathsf{vk}_j = \mathsf{VK}_j \cup [\mathsf{VK}'_{j,i}]_i$;

$\mathsf{Sign}(\mathsf{sk}_{j,i}, \tau, m)$: Given a signing key $\mathsf{sk}_{j,i} = [t, u, v, r, s]$, a message $m \in \mathbb{Z}_p$ and a public tag $\tau = (\tau_1, \tau_2, \tau_3)$, it outputs the signature (of m, by the j-th signer on the index (j, i)): $\sigma = \tau_1^{t+r+ms} \times \tau_2^u \times \tau_3^v \in \mathbb{G}_1$.

$\mathsf{AggrKey}((\mathsf{vk}_{j,i})_{j,i})$: Given verification keys $\mathsf{vk}_{j,i}$, it outputs the aggregated verification key $\mathsf{avk} = [\mathsf{avk}_j]_j$, with $\mathsf{avk}_j = \mathsf{VK}_j \cup [\mathsf{VK}'_{j,i}]_i$ for each j;

$\mathsf{AggrSign}(\tau, (\mathsf{vk}_{j,i}, m_{j,i}, \sigma_{j,i})_{j,i})$: Given tuples of verification key $\mathsf{vk}_{j,i}$, message $m_{j,i}$ and signature $\sigma_{j,i}$ all under the same tag τ, it outputs the signature $\sigma = \prod_{j,i} \sigma_{j,i} \in \mathbb{G}_1$ of the concatenation of the messages verifiable with $\mathsf{avk} \leftarrow \mathsf{AggrKey}((\mathsf{vk}_{j,i})_{j,i})$. Note that one needs to keep track of the indices of the $m_{j,i}$ in the concatenation;

$\mathsf{DerivSign}(\mathsf{avk}, \tau, \vec{M}, \sigma, \rho_{\tau\to\tau'})$: Given a signature σ on tag τ and a vector $\vec{M}$ (of tuples), and $\rho_{\tau\to\tau'}$ the randomization link between τ and another tag τ', it outputs $\sigma' = \sigma^{\rho_{\tau\to\tau'}}$;

$\mathsf{RandSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: The scheme being deterministic, it returns σ;

$\mathsf{VerifSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: Given a valid tag $\tau = (\tau_1, \tau_2, \tau_3)$, an aggregated verification key $\mathsf{avk} = [\mathsf{avk}_j]$ and a vector $\vec{M} = [\mathsf{m}_j]$, with both for each j, $\mathsf{avk}_j = \mathsf{VK}_j \cup [\mathsf{VK}'_{j,i}]_i$ and $\mathsf{m}_j = [m_{j,i}]_i$, and a signature σ, one checks if the following equality holds or not, where $n_j = \#\{\mathsf{VK}'_{j,i}\}$:

$$e(\sigma, \mathfrak{g}) = e\left(\tau_1, \prod_j \mathsf{VK}_{j,1}{}^{n_j} \times \prod_i \mathsf{VK}'_{j,i,1} \cdot \mathsf{VK}'_{j,i,2}{}^{m_{j,i}}\right) \times e\left(\tau_2, \prod_j \mathsf{VK}_{j,2}{}^{n_j}\right) \times e\left(\tau_3, \prod_j \mathsf{VK}_{j,3}{}^{n_j}\right).$$

In case of similar public keys in the aggregation (a unique index j), $\mathsf{avk} = \mathsf{VK} \cup [\mathsf{VK}'_i]_i$ and verification becomes, where $n = \#\{\mathsf{VK}'_i\}$,

$$e(\sigma, \mathfrak{g}) = e\left(\tau_1, \mathsf{VK}_1{}^n \times \prod_{i=1}^{n} \mathsf{VK}'_{i,1} \cdot \mathsf{VK}'_{i,2}{}^{m_i}\right) \times e\left(\tau_2, \mathsf{VK}_2{}^n\right) \times e\left(\tau_3, \mathsf{VK}_3{}^n\right).$$

Recall that the validity of the tag has to be verified, either with a proof of knowledge of the witness (as it will be the case in the ABC scheme, or with the proof $\pi = \mathsf{proof}(\tilde{\tau} : \tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}))$ (such as the one given in appendix F.2).

Security of the One-Time SqDH-based ART-Sign Scheme. As argued in the article [HPP19, Appendix C.5], the signature scheme defined above is unforgeable in the generic group model [Sho97], if signing queries are asked at most once per tag-index pair. About unlinkability, it relies on the DSqDH assumption, but between signatures that contain the same messages at the same shown indices (the same vector $\vec{M}$). Both security properties are stated below and proven in appendix C:

Theorem 16. *The One-Time SqDH-based ART-Sign is unforgeable with one signature only per index, for a given tag, even with adaptive corruptions of keys and tags, in the generic group model.*

Theorem 17. *The One-Time SqDH-based ART-Sign is unlinkable under the* DSqDH *and* DDH *assumptions.*

The Basic SqDH-based Anonymous Credential Scheme. The basic construction directly follows the instantiation of the above construction with the SqDH-based ART-Sign:

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$ be an asymmetric bilinear setting, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We then define $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, \mathcal{H})$, where $\mathcal{H}$ is an hash function in $\mathbb{G}_1$;

$\mathsf{CIKeyGen}(\mathsf{ID})$: Credential issuer CI with identity ID, generates its keys for n kinds of attributes

$$\mathsf{sk}_j = (\ \mathsf{SK}_j = [\ t_j,\ u_j,\ v_j\ ],\ \mathsf{SK}'_{j,i} = [\ r_{j,i},\ s_{j,i}\ ]_i\ ) \xleftarrow{\$} \mathbb{Z}_p^{3+2n},$$
$$\mathsf{vk}_j = (\ \mathsf{VK}_j = [\ \mathfrak{g}^{t_j}, \mathfrak{g}^{u_j}, \mathfrak{g}^{v_j}\ ],\ \mathsf{VK}'_{j,i} = [\ \mathfrak{g}^{r_{j,i}}, \mathfrak{g}^{s_{j,i}}\ ]_i\ ) \in \mathbb{G}_2^{3+2n}.$$

More keys for new attributes can be generated on-demand: by adding the pair $[r_{j,i}, s_{j,i}] \xleftarrow{\$} \mathbb{Z}_p^2$ to the secret key and $[\mathfrak{g}^{r_{j,i}}, \mathfrak{g}^{s_{j,i}}]$ to the verification key, the keys can works on $n+1$ kinds of attributes;

$\mathsf{UKeyGen}(\mathsf{id})$: User $\mathcal{U}$ with identity id, sets $h = \mathcal{H}(\mathsf{id}) \in \mathbb{G}_1^*$, generates its secret tag $\tilde{\tau} \xleftarrow{\$} \mathbb{Z}_p^*$ jointly with CA (to guarantee randomness) and computes $\tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$: $\mathsf{usk} = \tilde{\tau}$ and $\mathsf{uvk} = \tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2})$;

$(\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}, a_i), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a_i))$: User $\mathcal{U}$ with identity id and $\mathsf{uvk} = (\tau_1, \tau_2, \tau_3)$ asks to the credential issuer CI for a credential on the attribute a_i: $\sigma = \tau_1^{t+r_i+a_i s_i} \times \tau_2^u \times \tau_3^v \in \mathbb{G}_1$. The credential issuer uses the appropriate index i, making sure this is the first signature for this index;

$(\mathsf{CredShow}(\mathsf{usk}, (\mathsf{VK}_j, \mathsf{VK}'_{j,i}, a_{j,i}, \sigma_{j,i})_{j,i}), \mathsf{CredVerify}((\mathsf{VK}_j, \mathsf{VK}'_{j,i}, a_{j,i})_{j,i}))$:
First, user $\mathcal{U}$ randomizes his public key with a random $\rho \xleftarrow{\$} \mathbb{Z}_p^*$ into $\mathsf{uvk}' = (\tau_1^\rho, \tau_2^\rho, \tau_3^\rho)$, concatenates the keys $\mathsf{avk} = \cup_j([\mathsf{VK}_j] \cup [\mathsf{VK}'_{j,i}]_i)$, agregates $\sigma = \prod_{j,i} \sigma_{j,i} \in \mathbb{G}_1$, and adapts the signature $\sigma' = \sigma^\rho$. Then it sends the anonymous credential $(\mathsf{avk}, (a_{j,i})_{j,i}, \mathsf{uvk}', \sigma')$ to the verifier. The latter first checks the freshness of the credential with a proof of both ownership and validity of uvk' using a Schnorr-like interactive proof and then verifies the validity of the credential, with $n_j = \#\{\mathsf{VK}'_{j,i}\}$:

$$e(\sigma', \mathfrak{g}) = e\left(\tau_1, \prod_j \mathsf{VK}_{j,1}{}^{n_j} \times \prod_i \mathsf{VK}'_{j,i,1} \cdot \mathsf{VK}'_{j,i,2}{}^{a_{j,i}}\right) \times e\left(\tau_2, \prod_j \mathsf{VK}_{j,2}{}^{n_j}\right) \times e\left(\tau_3, \prod_j \mathsf{VK}_{j,3}{}^{n_j}\right).$$

We stress that for the unforgeability of the signature, generator h for each tag must be random, and so it is generated as $\mathcal{H}(\mathsf{id})$, with a hash function $\mathcal{H}$ in $\mathbb{G}_1$. This way, the credential issuers will automatically know the basis for each user. There is no privacy issue as this basis is randomized when used in an anonymous credential. Moreover, the user needs his secret key $\tilde{\tau}$ to be random. Therefore, he jointly generates $\tilde{\tau}$ with the Certification Authority (see appendix E). During the showing of a credential, the user has to make a fresh proof of knowledge of the witness for the validity of the tag. Again, in the security proof of unforgeability, one may need a rewinding, but only for the target alleged forgery.

In this construction, we can consider a polynomial number n of attributes per credential issuer, where a_i is associated to key $\mathsf{vk}_{j,i}$ of the Credential Issuer CI_j. Again, to keep the unforgeability of the signature, the credential issuer should provide at most one attribute per key $\mathsf{vk}_{j,i}$ for a given tag. At the showing time, for proving the ownership of k attributes (possibly from K different credential issuers), the users has to perform $k-1$ multiplications in $\mathbb{G}_1$ to aggregate the credentials into one, and 4 exponentiations in $\mathbb{G}_1$ for randomization, but just one element from $\mathbb{G}_1$ is sent, as anonymous credential, plus an interactive Schnorr-like proof of SqDH-tuple with knowledge of usk (see appendix F.1: 2 exponentiations in $\mathbb{G}_1$, 2 group elements from $\mathbb{G}_1$, and a scalar in $\mathbb{Z}_p$); whereas the verifier first has to perform 4 exponentiations and 2 multiplications in $\mathbb{G}_1$ for the proof of validity/knowledge of usk, and less than $3k$ multiplications and k exponentiations in $\mathbb{G}_2$, and 3 pairings to check the credential. While this is already better than [CL11], we can get a better construction (see appendix D).

We additionally remark that the aggregation σ is deterministic and can thus be kept for another showing of the same credentials. Only a new randomization of uvk into uvk' with the adaptation of σ into σ' is required for anonymity.

6 Traceable Anonymous Credentials

As the SqDH-based ART-Sign schemes provide computational unlinkability only, it opens the door of possible traceability in case of abuse, with anonymous but traceable tags.

The idea is that one can extend an EphemerId scheme with a modified GenTag algorithm and additional TraceId and JudgeId ones and use this traceable EphemerId to construct a traceable anonymous credentials, with similar properties as the ones defined for group signatures [BMW03]: an opener or a tracing authority can revoke anonymity (traceability), and publish the identity of the guilty, without being able to accuse an innocent (non-frameability or exculpability).

6.1 Traceable EphemerId

To help the reader, we extend the notations used in the anonymous credential to define the traceable EphemerId scheme:

Definition 18 (Traceable EphemerId). Based on an EphemerId scheme:

GenTag(param)**:** Given a public parameter param, it outputs the user-key pair (usk,uvk) and the tracing key utk;

TraceId(utk, uvk′)**:** Given the tracing key utk associated to uvk and a public key uvk′, it outputs a proof π of whether uvk $\sim$ uvk′ or not;

JudgeId(uvk, uvk′, π)**:** Given two public keys and a proof, the judge checks the proof π and outputs 1 if it is correct.

Construction. One can enhance our SqDH-based EphemerId scheme:

GenTag(param): Given a public parameter param, it randomly chooses a generator $h \xleftarrow{\$} \mathbb{G}_1^*$ and outputs $\mathsf{usk} = \tilde{\tau} \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathsf{uvk} = \tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$ and $\mathsf{utk} = \mathfrak{g}^{\tilde{\tau}}$;

TraceId(utk, uvk′): Given the tracing key utk associated to uvk $= (\tau_1, \tau_2, \tau_3)$ and a public key uvk′, it outputs a Groth-Sahai proof π (as shown in appendix F.3) that proves, in a zero-knowledge way, the existence of utk such that

$$e(\tau_1, \mathsf{utk}) = e(\tau_2, \mathfrak{g}) \qquad e(\tau_2, \mathsf{utk}) = e(\tau_3, \mathfrak{g}) \tag{1}$$

$$e(\tau_1', \mathsf{utk}) = e(\tau_2', \mathfrak{g}) \qquad e(\tau_2', \mathsf{utk}) = e(\tau_3', \mathfrak{g}); \tag{2}$$

JudgeId(uvk, uvk′, π): Given two public keys and a proof, the judge checks the proof π and outputs 1 if it is correct.

Correctness. The tracing key allows to check whether $\tau' \sim \tau$ or not: $e(\tau_1', \mathsf{utk}) = e(\tau_2', \mathfrak{g})$ and $e(\tau_2', \mathsf{utk}) = e(\tau_3', \mathfrak{g})$. If one already knows the tags are valid (SqDH tuples), this is enough to verify whether $e(\tau_1', \mathsf{utk}) = e(\tau_2', \mathfrak{g})$ holds or not. However we provide the complete proof in appendix F.3, as it is already quite efficient.The first equation (1) proves that utk is the good tracing key for uvk $= \tau$, and the second line (2) shows it applies to uvk′ $= \tau'$ too. It can be observed this can also be a proof of innocence of id with key uvk if the first equation (1) is satisfied while the second one is not.

The trapdoor allows traceability, and the soundness of the proof guarantees non-frameability or exculpability, as one cannot wrongly accuse a user.

6.2 Traceable Anonymous Credentials

For traceability in an anonymous credential scheme, we need an additional player: the *tracing authority*. During the user's key generation, this tracing authority will either be the certification authority, or a second authority, that also has to certify user's key uvk once it has received the tracing key utk.

We consider a non-interactive proof of tracing, produced by the TraceId algorithm and verified by anybody using the JudgeId algorithm. This proof could be interactive.

Non-frameability. In case of abuse of a credential σ under anonymous key uvk′, a tracing algorithm outputs the initial uvk and id, with a proof a correct tracing. A new security notion is quite important: *non-frameability*, which means that the tracing authority should not be able to declare guilty a wrong user: only correct proofs are accepted by the judge.

A successful adversary $\mathcal{A}$ against non-frameability is able to forge a valid credential σ^* under the key uvk* and a valid proof $\pi = \mathsf{TraceId}(\mathsf{utk}^*, \mathsf{uvk})$ for some honest user with identity id and key uvk which is not possible without breaking the unforgeability of the credential or the soundness of the proof. Hence, the tracing authority cannot frame a user and we obtain the first secure traceable anonymous credential scheme.

Acknowledgments

We warmly thank Olivier Sanders for fruitful discussions. This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud).

References

BDN18. Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 435–464. Springer, Heidelberg, December 2018.

BFI+10. Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 218–235. Springer, Heidelberg, June 2010.

BGLS03. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, Heidelberg, May 2003.

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

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

BMW03. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

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, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

CL11. Sébastien Canard and Roch Lescuyer. Anonymous credentials from (indexed) aggregate signatures. In Abhilasha Bhargav-Spantzel and Thomas Groß, editors, *DIM'11, Proceedings of the 2013 ACM Workshop on Digital Identity*, pages 53–62. ACM, 2011.

CL13. Sébastien Canard and Roch Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13*, pages 381–392. ACM Press, May 2013.

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

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

HPP19. Chloé Hébant, Duong Hieu Phan, and David Pointcheval. Linearly-homomorphic signatures and scalable mix-nets. Cryptology ePrint Archive, Report 2019/547, 2019. https://eprint.iacr.org/2019/547.

KL16. Nesrine Kaaniche and Maryline Laurent. Attribute-based signatures for supporting anonymous certification. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part I*, volume 9878 of *LNCS*, pages 279–300. Springer, Heidelberg, September 2016.

San20. Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 628–656. Springer, Heidelberg, May 2020.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.
Ver17. Damien Vergnaud. Comment on 'Attribute-Based Signatures for Supporting Anonymous Certification' by N. Kaaniche and M. Laurent (ESORICS 2016). *The Computer Journal*, 60(12):1801–1808, 2017.

A Canard-Lescuyer Scheme

In 2013, Canard and Lescuyer proposed a traceable attribute-based anonymous credential scheme [CL13], based on sanitizable signatures: "Protecting privacy by sanitizing personal data: a new approach to anonymous credentials".

The intuition consists in allowing the user to "sanitize" the global credentials issued by the credential issuer, in order to keep visible only the required attributes. Then for unlinkability, the signatures are encrypted under an ElGamal encryption scheme.

Unfortunately, in their scheme, the public key contains $g \xleftarrow{\$} \mathbb{G}_1$ and $\mathfrak{g} \xleftarrow{\$} \mathbb{G}_2$, and the ElGamal secret key is $\alpha \xleftarrow{\$} \mathbb{Z}_p$, the tracing key. The public encryption key is $h = g^\alpha$, but they also need $\mathfrak{h} = \mathfrak{g}^\alpha$ to be published for some verifications.

With this value $\mathfrak{h}$, anybody can break the semantic security of the ElGamal encryption, and then break the privacy of the anonymous credential.

B Classical Notations and Assumptions

All along the paper, κ is the security parameter. We will consider an asymmetric bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p (of length 2κ). The elements g and $\mathfrak{g}$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively and e is a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ into $\mathbb{G}_T$, that is non-degenerated and efficiently computable. This is usually called a *pairing*.

For the sake of clarity, elements of $\mathbb{G}_2$ will be in Fraktur font. In addition, in all the public-key cryptographic primitives, keys will implicitly include the global parameters and secret keys will include the public keys.

In an asymmetric bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$, or just in a simple group $\mathbb{G}$, we can define the following assumptions:

Definition 19 (Discrete Logarithm (DL) Assumption). In a group $\mathbb{G}$ of prime order p, it states that for any generator g, given $y = g^x$, it is computationally hard to recover x.

Definition 20 (Decisional Diffie-Hellman (DDH) Assumption). In a group $\mathbb{G}$ of prime order p, it states that the two following distributions are computationally indistinguishable:

$$\mathcal{D}_{\mathsf{dh}} = \{(g, g^x, h, h^x); g, h \xleftarrow{\$} \mathbb{G}, x, \xleftarrow{\$} \mathbb{Z}_p\}$$
$$\mathbb{G}^4_{\$} = \{(g, g^x, h, h^y); g, h \xleftarrow{\$} \mathbb{G}, x, y, \xleftarrow{\$} \mathbb{Z}_p\}.$$

C Proofs of Theorems

C.1 Proof of Theorem 13

Let $\mathcal{A}$ be an adversary against the unforgeability of our anonymous credential scheme. We build an adversary $\mathcal{B}$ against the unforgeability of the RT-Sign. We stress that our proof is in the certified key model: even for the corrupted players, the simulator knows the secret keys, as they can be extracted at the certification time. Our adversary $\mathcal{B}$ runs the unforgeability security game of the RT-Sign, and answers the oracle queries asked by $\mathcal{A}$ as follows:

- $\mathcal{O}$HCI(ID): If ID $\in$ HCI $\cup$ CCI, $\mathcal{B}$ outputs $\perp$. Otherwise, it adds ID $\in$ HCI, asks the query $\mathcal{O}$Keygen() and forwards the answer to $\mathcal{A}$;

- $\mathcal{O}\mathsf{CCI}(\mathsf{ID}, \mathsf{vk})$: If $\mathsf{ID} \notin \mathsf{HCI} \cup \mathsf{CCI}$, $\mathcal{B}$ adds $\mathsf{ID} \in \mathsf{CCI}$. Otherwise, if $\mathsf{ID} \in \mathsf{HCI}$ with keys $(\mathsf{sk}, \mathsf{vk})$, it moves ID from HCI to CCI. It then asks the query $\mathcal{O}\mathsf{Corrupt}(\mathsf{vk})$ and forwards the answer to $\mathcal{A}$;
- $\mathcal{O}\mathsf{HU}(\mathsf{id})$: If $\mathsf{id} \in \mathsf{HU} \cup \mathsf{CU}$, $\mathcal{B}$ outputs $\perp$. Otherwise, it adds $\mathsf{id} \in \mathsf{HU}$, asks the query $\mathcal{O}\mathsf{GenTag}()$ and forwards the answer to $\mathcal{A}$;
- $\mathcal{O}\mathsf{CU}(\mathsf{id}, \mathsf{uvk})$: If $\mathsf{id} \notin \mathsf{HU} \cup \mathsf{CU}$, $\mathcal{B}$ adds $\mathsf{id} \in \mathsf{CU}$. Otherwise, if $\mathsf{id} \in \mathsf{HU}$ with keys $(\mathsf{usk}, \mathsf{uvk})$, it moves id from HU to CU, asks the query $\mathcal{O}\mathsf{CorruptTag}(\mathsf{uvk})$ and forwards the answer to $\mathcal{A}$;
- $\mathcal{O}\mathsf{ObtIss}(\mathsf{id}, \mathsf{ID}, a)$: If $\mathsf{id} \notin \mathsf{HU}$ or $\mathsf{ID} \notin \mathsf{HCI}$, $\mathcal{B}$ outputs $\perp$. Otherwise, id is associated to $(\mathsf{usk}, \mathsf{uvk})$ and ID is associated to $(\mathsf{sk}, \mathsf{vk})$. Then $\mathcal{B}$ asks the query $\mathcal{O}\mathsf{Sign}(\mathsf{vk}, \mathsf{uvk}, a)$, adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$ and outputs σ.
- $\mathcal{O}\mathsf{Obtain}(\mathsf{id}, \mathsf{ID}, a)$: If $\mathsf{id} \notin \mathsf{HU}$ or $\mathsf{ID} \notin \mathsf{CCI}$, $\mathcal{B}$ outputs $\perp$. Otherwise, id is associated to $(\mathsf{usk}, \mathsf{uvk})$ and ID is associated to vk. In our (non-interactive) construction, the adversary additionally provides a signature σ, that is first checked by $\mathcal{B}$, that later adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$;
- $\mathcal{O}\mathsf{Issue}(\mathsf{id}, \mathsf{ID}, a)$: If $\mathsf{id} \notin \mathsf{CU}$ or $\mathsf{ID} \notin \mathsf{HCI}$, $\mathcal{B}$ outputs $\perp$. Otherwise, id is associated to uvk and ID is associated to $(\mathsf{sk}, \mathsf{vk})$. Then $\mathcal{B}$ runs $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{uvk}, a)$ and adds (ID, a) to $\mathsf{Att}[\mathsf{id}]$ and (ID, a, σ) to $\mathsf{Cred}[\mathsf{id}]$;
- $\mathcal{O}\mathsf{Show}(\mathsf{id}, (\mathsf{ID}_j, a_j)_j)$: If $\mathsf{id} \notin \mathsf{HU}$ or $\{(\mathsf{ID}_j, a_j)\}_j \not\subset \mathsf{Att}[\mathsf{id}]$, $\mathcal{B}$ outputs $\perp$. Otherwise, id is associated to $(\mathsf{usk}, \mathsf{uvk})$ and each ID_j is associated to $(\mathsf{sk}_j, \mathsf{vk}_j)$. Furthermore, for each (ID_j, a_j), there is σ_j such that $(\mathsf{ID}_j, a_j, \sigma_j) \in \mathsf{Cred}[\mathsf{id}]$. Then $\mathcal{B}$ first randomizes the key uvk with $(\mathsf{uvk}', \rho) = \mathcal{E}.\mathsf{RandTag}(\mathsf{uvk})$, and adapts the secret key $\mathsf{usk}' = \mathcal{E}.\mathsf{DerivWitness}(\mathsf{usk}, \rho)$. From the obtained credentials σ_j, it adapts $(\sigma'_j)_j = \mathsf{S}^{\mathrm{rt}}.\mathsf{DerivSign}((\mathsf{vk}_j)_j, \mathsf{uvk}, (a_j, \sigma_j)_j, \rho)$, and randomizes it: $(\sigma''_j)_j = \mathsf{S}^{\mathrm{rt}}.\mathsf{RandSign}((\mathsf{vk}_j)_j, \mathsf{uvk}', (a_j, \sigma'_j)_j)$. $\mathcal{B}$ then outputs $((\mathsf{vk}_j, a_j, \sigma''_j)_j, \mathsf{uvk}')$ and makes the $\mathcal{E}.\mathsf{ProveKTag}(\mathsf{usk}')$ part of the interactive proof of ownership.

Eventually, the adversary $\mathcal{A}$ runs a showing for $(\mathsf{vk}_j, a_j)_j$, with a credential $((\mathsf{vk}_j, a_j, \sigma^*_j)_j, \mathsf{uvk}^*)$ and a proof of knowledge of usk^* associated to uvk^*: in case of success, $\mathcal{B}$ outputs the signature $\vec{\sigma^*} = (\sigma^*_j)_j$ of $\vec{a} = (a_j)_j$ for the keys $\vec{\mathsf{vk}} = (\mathsf{vk}_j)_j$ under the tag uvk^*.

In case of validity of the showing, except with negligible probability,

- from the knowledge-soundness of the $\mathsf{EphemerId}$ scheme, with a valid final showing and proof of knowledge, one can use once the extractor to get usk^*. This proves the validity of the tag uvk^*;
- from the unforgeability of the signature with randomizable tags on vectors, all the attributes a_j's have been signed for vk_j and a common $\mathsf{uvk} \sim \mathsf{uvk}^*$, such that there is $\mathsf{id} \in \mathsf{CU}$, associated to uvk. These individual credentials have thus been issued either by the adversary on behalf of a corrupted credential issuer $\mathsf{ID}_j \in \mathsf{CCI}$ or from an oracle query to ID_j for id.

This is thus a legitimate showing with overwhelming probability: $\mathcal{B}$ win with negligible probability. Hence, this is the same for the adversary $\mathcal{A}$.

C.2 Proof of Theorem 14

From the unlinkability of the $\mathsf{RT\text{-}Sign}$, the tuple $((\mathsf{vk}_j, a_j, \sigma''_j)_j, \mathsf{uvk}')$ does not leak any information about the initial tag. Hence, a credential does not leak any information about uvk_b. In addition, if the proof of knowledge of the witness is zero-knowledge, it does not leak any information about uvk_b either.

C.3 Proof of Theorem 16

As argued in [HPP19], when the bases of the tags are *random*, even if the exponents are known, the signature on messages $(g, g^{m_1}, \ldots, g, g^{m_n}) \in \mathbb{G}_1^{2n}$ is an unforgeable linearly-homomorphic

signature. While [HPP19] was signing vectors in $\mathbb{G}_1$, unforgeability also holds when $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$ is known. This means it is only possible to linearly combine signatures with the same tag. As issued signatures are on pairs (g, g^{m_i}), under a different pair of keys $\mathsf{sk}_{j,i}$ for each such signed pair (whether they are from the same global signing key SK_j or not, as we exclude repetitions for an index), which can be seen as tuples $(1, 1, \ldots, g, g^{m_i}, \ldots, 1, 1)$, completed with $1 = g^0$: all the pairs (g, g^{m_i}) have been signed under the same tag. This proves unforgeability, even with corruptions of the tags, but without repetitions of tag-index. One can also consider corruptions of the signing keys, as they are all independent: one just needs to guess under which key will be generated the forgery.

C.4 Proof of Theorem 17

As already noticed, the tags are randomizable among all the square Diffie-Hellman triples with the same exponent, and for any pair of tags $(\tilde{\tau}_i, \tau_i) \leftarrow \mathsf{GenTag}(1^\kappa)$, for $i = 0, 1$, when randomized into τ_i' respectively, the distributions $(\tau_0, \tau_1, \tau_0', \tau_1')$ and $(\tau_0, \tau_1, \tau_1', \tau_0')$ are indistinguishable from $\mathbb{G}_\$^{12}$ under the DSqDH and DDH assumptions, as shown in Theorem 3. For any avk and $\vec{m}$, the signatures are deterministic and unique for a tag τ, so they are functions of $(\mathsf{avk}, \tau, \vec{m})$. Then, using the signing keys, one can get that the distributions $(\vec{m}, \mathsf{avk}, \tau_0, \sigma_0, \tau_1, \sigma_1, \tau_0', \sigma_0', \tau_1', \sigma_1)$ and $(\vec{m}, \mathsf{avk}, \tau_0, \sigma_0, \tau_1, \sigma_1, \tau_1', \sigma_1', \tau_0', \sigma_0)$ are also indistinguishable under the DSqDH and DDH assumption. No need of randomization of the signatures.

D Bounded SqDH-Based ART-Sign Schemes

The construction from Section 5 limits to one-time signatures: only one signature can be generated for a given tag-index, otherwise signatures can be later forged on any message for this index, by linearity [HPP19, Appendix C.5]: the vector space spanned by (g, g^m) (in case of just one signature issued for one index) is just $(g^\alpha, g^{\alpha m})$ and the ratio of the exponents is the constant m; on the other hand, the vector space spanned by (g, g^m) and $(g, g^{m'})$ (in case of two signatures issued for one index) is $\mathbb{G}_1 \times \mathbb{G}_1$, and then any ratio can be achieved.

This will be enough for our ABC application, as one usually has one attribute value for a specific kind of information (age, city, diploma, etc), but in practice this implies the signer to either keep track of all the indices already signed for one tag or to sign all the messages at once. We provide another kind of combinations, that could be applied on our SqDH signature scheme that will have interesting application to an ABC scheme.

D.1 Bounded SqDH-based ART-Sign Scheme 1

Description. We propose here an alternative where the limitation is on the total number n of messages signed for each tag by each signer:

$\mathsf{Setup}(1^\kappa)$: It extends the above Ephemerld-setup with the set of messages $\mathcal{M} = \mathbb{Z}_p$;

$\mathsf{Keygen}(\mathsf{param}, n)$: Given the public parameters param and a length n, it outputs the signing and verification keys

$$\mathsf{sk}_j = [\ t_j,\ u_j,\ v_j,\ s_{j,1},\ \ldots,\ s_{j,2n-1}\ ] \xleftarrow{\$} \mathbb{Z}_p^{2n+2},$$
$$\mathsf{vk}_j = \mathfrak{g}^{\mathsf{sk}_j} = [\ T_j, U_j, V_j, S_{j,1}, \ldots, S_{j,2n-1}\ ] \in \mathbb{G}_2^{2n+2}.$$

$\mathsf{Sign}(\mathsf{sk}_j, \tau, m)$: Given a signing key $\mathsf{sk}_j = [t, u, v, s_1, \ldots, s_{2n-1}]$, a message $m \in \mathbb{Z}_p$ and a public tag $\tau = (\tau_1, \tau_2, \tau_3)$, it outputs the signature

$$\sigma = \tau_1^{t+\sum_1^{2n-1} s_\ell m^\ell} \times \tau_2^u \times \tau_3^v \in \mathbb{G}_1.$$

$\mathsf{AggrKey}(\{\mathsf{vk}_j\}_j)$: Given verification keys vk_j, it outputs the aggregated verification key $\mathsf{avk} = [\mathsf{vk}_j]_j$;

$\mathsf{AggrSign}(\tau, (\mathsf{vk}_j, m_{j,i}, \sigma_{j,i})_{j,i})$: Given tuples of verification key vk_j, message $m_{j,i}$ and signature $\sigma_{j,i}$ all under the same tag τ, it outputs the signature $\sigma = \prod_{j,i} \sigma_{j,i} \in \mathbb{G}_1$ of the concatenation of the messages verifiable with $\mathsf{avk} \leftarrow \mathsf{AggrKey}(\{\mathsf{vk}_j\}_j)$;

$\mathsf{DerivSign}(\mathsf{avk}, \tau, \vec{M}, \sigma, \rho_{\tau\to\tau'})$: Given a signature σ on tag τ and a message-set $\vec{M}$, and $\rho_{\tau\to\tau'}$ the randomization link between τ and another tag τ', it outputs $\sigma' = \sigma^{\rho_{\tau\to\tau'}}$;

$\mathsf{RandSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: The scheme being deterministic, it returns σ;

$\mathsf{VerifSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: Given a valid tag $\tau = (\tau_1, \tau_2, \tau_3)$, an aggregated verification key $\mathsf{avk} = [\mathsf{vk}_j]_j$ and a message-set $\vec{M} = [\mathsf{m}_j]_j$, with for each j, $\mathsf{m}_j = [m_{j,i}]_i$, and a signature σ, one checks if the following equality holds or not, where $n_j = \#\{m_{j,i}\}$:

$$e(\sigma, \mathfrak{g}) = e\left(\tau_1, \prod_j T_j^{n_j} \times \prod_{\ell=1}^{2n-1} S_{j,\ell}^{\sum_i m_{j,i}^\ell}\right) \times e\left(\tau_2, \prod_j U_j^{n_j}\right) \times e\left(\tau_3, \prod_j V_j^{n_j}\right)$$

Recall that the validity of the tag has to be verified, as for the other version.

Security Analysis. The linear homomorphism of the signature from [HPP19, Appendix C.5] still allows combinations. But when the number of signing queries is at most $2n$ per tag, the verification of the signature implies 0/1 coefficients only:

Theorem 21. *The bounded SqDH-based ART-Sign is unforgeable with a bounded number of signing queries per tag, even with adaptive corruptions of keys and tags, in both the generic group model and the random oracle model.*

Proof. As argued in [HPP19, Appendix C.5] and recalled in Theorem 4, when the bases of the tags are random, even if the exponents are known, the signature that would have signed messages $(g^{m^1}, \ldots, g^{m^{2n-1}})$, for $m \in \mathbb{Z}_p$, is an unforgeable linearly-homomorphic signature. This means it is only possible to linearly combine signatures with the same tag. We fix the limit to n signatures σ_i queried on distinct messages m_i, for $i = 1, \ldots, n$ under vk_j: one can derive the signature $\sigma = \prod \sigma_i^{\alpha_i}$ on $\left(g^{\sum_i \alpha_i m_i^1}, \ldots, g^{\sum_i \alpha_i m_i^{2n-1}}\right)$. Whereas the forger claims this is a signature on $\left(g^{\sum_i a_i^1}, \ldots, g^{\sum_i a_i^n}\right)$, on $n_j \leq n$ values $a_1, \ldots, a_{n_j}$, as one cannot combine more than n messages. Because of the constraint on τ_2, we additionally have $\sum \alpha_i = n_j \bmod p$:

$$\sum_{i=1}^{n} \alpha_i m_i^\ell = \sum_{i=1}^{n_j} a_i^\ell \bmod p \qquad \text{for } \ell = 0, \ldots, 2n-1$$

Let us first move on the left hand side the elements $a_k \in \{m_i\}$, with only $n' \leq n_j$ new elements, we assume to be the first ones, and we note $\beta_i = \alpha_i$ if $m_i \notin \{a_k\}$ and $\beta_i = \alpha_i - 1$ if $m_i \in \{a_k\}$: $\sum_{i=1}^{n} \beta_i m_i^\ell = \sum_{i=1}^{n'} a_i^\ell \bmod p$, for $\ell = 0, \ldots, 2n-1$. We thus have the system

$$\sum_{i=1}^{n} \beta_i m_i^\ell + \sum_{i=1}^{n'} \gamma_i a_i^\ell = 0 \bmod p \qquad \text{for } \ell = 0, \ldots, 2n-1, \text{ with } \gamma_i = -1$$

This is a system of $2n$ equations with at most $n + n' \leq 2n$ unknown values β_i's and γ_i's, and the Vandermonde matrix is invertible: $\beta_i = 0$ and $\gamma_i = 0$ for all index i. As a consequence, the vector $(\alpha_i)_i$ only contains 0 or 1 components.

This proves unforgeability, even with corruptions of the tags, but with a number of signed messages bounded by n. One can also consider corruptions of the signing keys, as they are all independent: one just needs to guess under which key will be generated the forgery.

About unlinkability, it relies on the DSqDH assumption, with the same proof as the previous scheme:

Theorem 22. *The bounded SqDH-based ART-Sign, is unlinkable.*

The Compact SqDH-based Anonymous Credential Scheme. Instead of having a specific key $\mathsf{VK}'_{j,i}$ for each family of attributes $a_{j,i}$, and thus limiting to one issuing per family of attributes for each user, we can use the bounded SqDH-based ART-Sign, with free-text attributes: we consider $2n-1$ keys, where n is the maximum number of attributes issued for one user by a credential issuer, whatever the attributes are:

$\mathsf{Setup}(1^\kappa)$: Given a security parameter κ, let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e)$ be an asymmetric bilinear setting, where g and $\mathfrak{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We then define $\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathfrak{g}, e, \mathcal{H})$, where $\mathcal{H}$ is an hash function in $\mathbb{G}_1$;

$\mathsf{CIKeyGen}(\mathsf{ID})$: Credential issuer CI with identity ID, generates its keys for n maximum attributes per user

$$\mathsf{sk}_j = [\ t_j,\ u_j,\ v_j,\ s_{j,1},\ \ldots,\ s_{j,2n-1}\ ] \xleftarrow{\$} \mathbb{Z}_p^{2n+2},$$
$$\mathsf{vk}_j = \mathfrak{g}^{\mathsf{sk}_j} = [\ T_j, U_j, V_j, S_{j,1}, \ldots, S_{j,2n-1}\ ] \in \mathbb{G}_2^{2n+2}.$$

$\mathsf{UKeyGen}(\mathsf{id})$: User $\mathcal{U}$ with identity id, sets $h = \mathcal{H}(\mathsf{id}) \in \mathbb{G}_1^*$, , generates its secret tag $\tilde{\tau} \xleftarrow{\$} \mathbb{Z}_p^*$ jointly with CA (to guarantee randomness) and computes $\tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$: $\mathsf{usk} = \tilde{\tau}$ and $\mathsf{uvk} = \tau = (h, h^{\tilde{\tau}}, h^{\tilde{\tau}^2})$;

$(\mathsf{CredObtain}(\mathsf{usk}, \mathsf{vk}, a), \mathsf{CredIssue}(\mathsf{uvk}, \mathsf{sk}, a))$: User $\mathcal{U}$ with identity id and $\mathsf{uvk} = (\tau_1, \tau_2, \tau_3)$ asks to the credential issuer CI for a credential on the attribute a: $\sigma = \tau_1^{t+\sum_{\ell=1}^{2n-1} s_\ell a^\ell} \times \tau_2^u \times \tau_3^v \in \mathbb{G}_1$. Note that $a \in \mathbb{Z}_p^*$, so it can be a hash value of an attribute represented by an arbitrary bit string;

$(\mathsf{CredShow}(\mathsf{usk}, (\mathsf{vk}_j, a_{j,i}, \sigma_{j,i})_{j,i}), \mathsf{CredVerify}((\mathsf{vk}_j, a_{j,i})_{j,i})$: First, a user $\mathcal{U}$ randomizes his public key with a random $\rho \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathsf{uvk}' = (\tau_1^\rho, \tau_2^\rho, \tau_3^\rho)$, concatenates the keys $\mathsf{avk} = \cup_j[\mathsf{vk}_j]$, and the signatures $\sigma = \prod_{j,i} \sigma_{j,i} \in \mathbb{G}_1$. It then adapts the signature $\sigma' = \sigma^\rho$. It eventually sends the anonymous credential $(\mathsf{avk}, (a_{j,i})_{j,i}, \mathsf{uvk}', \sigma')$ to the verifier. The latter first checks the freshness of the credential with a proof of ownership and validity of uvk' using a Schnorr-like interactive proof and then verifies the validity of the credential: with $n_j = \#\{a_{j,i}\}$:

$$e(\sigma', \mathfrak{g}) = e\left(\tau_1, \prod_j T_j^{n_j} \prod_{\ell=1}^{2n-1} S_{j,\ell}^{\sum_i a_{j,i}^\ell}\right) \times e\left(\tau_2, \prod_j U_j^{n_j}\right) \times e\left(\tau_3, \prod_j V_j^{n_j}\right)$$

Again, we stress that for the unforgeability of the signature, generator h for each tag and $\tilde{\tau}$ must be random. And the credential issuer should provide at most n attributes per user, even if in this construction, we can consider an exponential number N of attributes per credential issuer, as $a_{j,i}$ is any scalar in $\mathbb{Z}_p^*$. More concretely, $a_{j,i}$ can be given as the output of a hash function into $\mathbb{Z}_p$ from any bitstring. At the showing time, for proving the ownership of k attributes (possibly from K different credential issuers), the users has to perform $k-1$ multiplications in $\mathbb{G}_1$ to aggregate the credentials into one, and 4 exponentiations in $\mathbb{G}_1$ for randomization, but just one group element for $\mathbb{G}_1$ is sent, as anonymous credential, plus an interactive Schnorr-like proof of SqDH-tuple with knowledge of usk (see appendix F.1: 2 exponentiations in $\mathbb{G}_1$, 2 group elements from $\mathbb{G}_1$, and a scalar in $\mathbb{Z}_p$); whereas the verifier first has to perform 4 exponentiations and 2 multiplications in $\mathbb{G}_1$ for the proof of validity/knowledge of usk, and less than $2n \cdot (K + 3k)$ multiplications in $\mathbb{G}_2$, $2n \cdot k$ exponentiations in $\mathbb{G}_2$ and 3 pairings to check the credential.

In the particular case of just one credential issuer with verification key $\mathsf{vk} = (T, U, V, [S_i]_{i=1}^{2n-1})$, the verification of the credential σ on the k attributes $\{a_i\}$ just consists of

$$e(\sigma, \mathfrak{g}) = e\left(\tau_1, T^k \prod_{\ell=1}^{2n-1} S_\ell^{\sum_i a_i^\ell}\right) \times e\left(\tau_2, U^k\right) \times e\left(\tau_3, V^k\right).$$

The communication is of constant size (one group element in $\mathbb{G}_1$). We stress that n is just a limit of the maximal number of attributes issued by the credential issuer for one user but the universe of the possible attributes is exponentially large, and there is no distinction between the families of attributes.

D.2 Bounded SqDH-based ART-Sign Scheme 2

We can slightly reduce the parameters of the bounded SqDH-based ART-Sign, but with some limitations on the number of attributed to be signed. It relies on a hash function, modelled as a random oracle in the security analysis.

Description. We propose here a second version, still with the limitation on the total number of messages signed for each tag, but the public keys are twice smaller:

$\mathsf{Setup}(1^\kappa)$: It extends the above Ephemerld-setup with the set of messages $\mathcal{M} = \{0,1\}^*$, but also a hash function $\mathcal{H}$ into $\mathbb{Z}_p$;

$\mathsf{Keygen}(\mathsf{param}, n)$: Given the public parameters param and a length n, it outputs the signing and verification keys

$$\mathsf{sk}_j = [\ t_j,\ u_j,\ v_j,\ s_{j,1},\ \ldots,\ s_{j,n}\ ] \xleftarrow{\$} \mathbb{Z}_p^{n+3},$$
$$\mathsf{vk}_j = \mathfrak{g}^{\mathsf{sk}_j} = [\ T_j, U_j, V_j, S_{j,1}, \ldots, S_{j,n}\ ] \in \mathbb{G}_2^{n+3}.$$

$\mathsf{Sign}(\mathsf{sk}_j, \tau, m)$: Given a signing key $\mathsf{sk}_j = [t, u, v, s_1, \ldots, s_n]$, a message $m \in \mathbb{Z}_p$ and a public tag $\tau = (\tau_1, \tau_2, \tau_3)$, it outputs the signature

$$\sigma = \tau_1^{t+\sum_{\ell=1}^n s_\ell \mathcal{H}(m)^\ell} \times \tau_2^u \times \tau_3^v.$$

$\mathsf{AggrKey}(\{\mathsf{vk}_j\}_j)$: Given verification keys vk_j, it outputs the aggregated verification key $\mathsf{avk} = [\mathsf{vk}_j]_j$;

$\mathsf{AggrSign}(\tau, (\mathsf{vk}_j, m_{j,i}, \sigma_{j,i})_{j,i})$: Given tuples of verification key vk_j, message $m_{j,i}$ and signature $\sigma_{j,i}$ all under the same tag τ, it outputs the signature $\sigma = \prod_{j,i} \sigma_{j,i}$ of the concatenation of the messages verifiable with $\mathsf{avk} \leftarrow \mathsf{AggrKey}(\{\mathsf{vk}_j\}_j)$;

$\mathsf{DerivSign}(\mathsf{avk}, \tau, \vec{M}, \sigma, \rho_{\tau\to\tau'})$: Given a signature σ on tag τ and a message-set $\vec{M}$, and $\rho_{\tau\to\tau'}$ the randomization link between τ and another tag τ', it outputs $\sigma' = \sigma^{\rho_{\tau\to\tau'}}$;

$\mathsf{RandSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: The scheme being deterministic, it returns σ;

$\mathsf{VerifSign}(\mathsf{avk}, \tau, \vec{M}, \sigma)$: Given a valid tag $\tau = (\tau_1, \tau_2, \tau_3)$, an aggregated verification key $\mathsf{avk} = [\mathsf{vk}_j]_j$ and a message-set $\vec{M} = [\mathsf{m}_j]_j$, with for each j, $\mathsf{m}_j = [m_{j,i}]_i$, and a signature σ, one checks if the following equality holds or not, where $n_j = \#\{m_{j,i}\}$:

$$e(\sigma, \mathfrak{g}) = e\left(\tau_1, \prod_j T_j^{n_j} \times \prod_{\ell=1}^n S_{j,\ell}^{\sum_i \mathcal{H}(m_{j,i})^\ell}\right) \times e\left(\tau_2, \prod_j U_{j,2}^{n_j}\right) \times e\left(\tau_3, \prod_j V_{j,3}^{n_j}\right)$$

We also recall that the validity of the tag has to be verified, as before, for the signature to be considered valid.

Security Analysis. The linear homomorphism of the signature from [HPP19, Appendix C.5] still allows combinations. But when the number of signing queries is at most n per tag, the verification of the signature implies 0/1 coefficients only, with overwhelming probability:

Theorem 23. *The bounded SqDH-based ART-Sign defined above is unforgeable with a bounded number of signing queries per tag, even with adaptive corruptions of keys and tags, in both the generic group model and the random oracle model, as soon as $q_{\mathcal{H}}^n \ll p$, where $q_{\mathcal{H}}$ is the number of hash queries and p the order of the group (the output of the hash function).*

Proof. As argued in [HPP19, Appendix C.5], when the bases of the tags are random, even if the exponents are known, the signature that would have signed messages $(g^{m^1}, \ldots, g^{m^n})$, for $m \in \mathbb{Z}_p$, is an unforgeable linearly-homomorphic signature. This means it is only possible to linearly combine signatures with the same tag: from up to n signatures σ_i on distinct messages m_i, for $i = 1, \ldots, n$ under vk_j, one can derive the signature $\sigma = \prod \sigma_i^{\alpha_i}$ on $\left(g^{\sum_i \alpha_i m_i^1}, \ldots, g^{\sum_i \alpha_i m_i^n}\right)$. Whereas the forger claims this is a signature on $\left(g^{\sum_i a_i^1}, \ldots, g^{\sum_i a_i^n}\right)$, on n_j values $a_1, \ldots, a_{n_j}$. Because of the constraint on τ_2, we have $\sum \alpha_i = n_j \bmod p$:

$$\sum_{i=1}^{n} \alpha_i m_i^\ell = \sum_{i=1}^{n_j} a_i^\ell \bmod p \qquad \text{for } \ell = 0, \ldots, n$$

Let us first move on the left hand side the elements $a_k \in \{m_i\}$, with only $n' \leq n_j$ new elements, we assume to be the first ones, and we note $\beta_i = \alpha_i$ if $m_i \notin \{a_k\}$ and or $\beta_i = \alpha_i - 1$ if $m_i \in \{a_k\}$:

$$\sum_{i=1}^{n} \beta_i m_i^\ell = \sum_{i=1}^{n'} a_i^\ell \bmod p \qquad \text{for } \ell = 0, \ldots, n$$

Our goal is to prove that $n' = 0$ and the α_i's are only 0 or 1.

So, first, let us assume that $n' = 0$: there is no new element. The matrix $(m_i^\ell)_{i,\ell}$, for $i = 1, \ldots, n$ and $\ell = 0, \ldots, n-1$ is a Vandermonde matrix, that is invertible: hence the unique possible vector (β_i) is the zero-vector. As a consequence, the vector $(\alpha_i)_i$ only contains 0 or 1 components.

Now, we assume $n' = 1$: there is exactly one element $a_1 \notin \{m_i\}$. We can move it on the left side:

$$\beta_0 a_1^\ell + \sum_{i=1}^{n} \beta m_i^\ell = 0 \bmod p \qquad \text{for } \ell = 0, \ldots, n, \text{ with } \beta_0 = -1$$

Again, the matrix $(m_i^\ell)_{i,\ell}$, for $i = 0, \ldots, n$ where we denote $m_0 = a_1$, and $\ell = 0, \ldots, n$, is a Vandermonde matrix, that is invertible: hence the unique possible vector (β_i) is the zero-vector, which contradicts the fact that $\beta_0 = -1$.

Eventually, we assume $n' > 1$: there are at least two elements $a_k \notin \{m_i\}$. We can move a_1 on the left side:

$$\beta_0 a_1^\ell + \sum_{i=1}^{n} \beta m_i^\ell = \sum_{i=2}^{n'} a_i^\ell \bmod p \qquad \text{for } \ell = 0, \ldots, n, \text{ with } \beta_0 = -1$$

Again, because of the invertible matrix, for the $n' - 1$ elements on the right hand side, there is a unique possible vector (β_i), and the probability for $\beta_0 = -1$ is negligible, as the new elements a_k are random (if they are issued from a hash value): probability $1/p$ for each possible choice on the $n' - 1 < n$ attributes on the right hand side. Hence, as soon as $q_{\mathcal{H}}^n \ll p$, the probability for a combination to allow $\beta_0 = -1$ is negligible.

As a conclusion, one can only combine initial messages with a weight 1 (or 0). This proves unforgeability, even with corruptions of the tags, but with a number of signed messages bounded by n, and random messages (issued from a hash function). One can also consider corruptions of the signing keys, as they are all independent: one just needs to guess under which key will be generated the forgery.

Unlinkability remains unchanged.

E Joint Generation of Square Diffie-Hellman Tuples

As already explained, for the unlinkability property to hold in the anonymous credential protocol, we need the user secret key $\mathsf{usk} = \tilde{\tau}$ random. Of course, this could be done with generic two-party computation, between the user and the Certification Authority.

- The user chooses $\tilde{\tau}_1 \xleftarrow{\$} \mathbb{Z}_p$ and computes $(A_1 = h^{\tilde{\tau}_1}, B_1 = A_1^{\tilde{\tau}_1})$.
- On its side, the Certification Authority chooses $\tilde{\tau}_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$A = A_1 \cdot h^{\tilde{\tau}_2} = h^{\tilde{\tau}_1+\tilde{\tau}_2} \qquad B = B_1 \cdot (A_1^2 \cdot h^{\tilde{\tau}_2})^{\tilde{\tau}_2} = A_1^{\tilde{\tau}_1} \cdot A_1^{2\tilde{\tau}_2} h^{\tilde{\tau}_2^2} = h^{(\tilde{\tau}_1+\tilde{\tau}_2)^2}.$$

  It then sends and certifies $\tau = (h, A, B)$ together with $\tilde{\tau}_2$ so that the user can compute $\tilde{\tau} = \tilde{\tau}_1 + \tilde{\tau}_2$.

F Zero-Knowledge Proofs

F.1 Zero-Knowledge Proof for Square Diffie-Hellman Tuples

During both the certification of the tag τ and the showing protocol, the user must provide a proof of validity of the SqDH tuple, in an extractable way, as this must also be a proof of knowledge.

As an SqDH-tuple $(\tau_1 = h, \tau_2 = h^{\tilde{\tau}}, \tau_3 = h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$ is a Diffie-Hellman tuple $(\tau_1, \tau_2, \tau_2, \tau_3)$, one can use a Schnorr-like proof:

- The prover chooses a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$, and sets and sends $U \leftarrow \tau_1^r$, $V \leftarrow \tau_2^r$;
- The verifier chooses a random challenge $e \xleftarrow{\$} \{0,1\}^\kappa$;
- The prover sends back the response $s = e\tilde{\tau} + r \bmod p$;
- The verifier checks whether both $\tau_1^s = \tau_2^e \times U$ and $\tau_2^s = \tau_3^e \times V$.

This provides an interactive zero-knowledge proof of knowledge of the witness $\tilde{\tau}$ that (τ_1, τ_2, τ_3) is an SqDH-tuple.

F.2 Groth-Sahai Proof for Square Diffie-Hellman Tuples

If one just needs a proof of validity of the tuple, this is possible, using the Groth-Sahai methodology [GS08], to provide a non-interactive proof of Square Diffie-Hellman tuple: in the asymmetric pairing setting, one sets a reference string $(\mathfrak{v}_{1,1}, \mathfrak{v}_{1,2}, \mathfrak{v}_{2,1}, \mathfrak{v}_{2,2}) \in \mathbb{G}_2^4$, such that $(\mathfrak{v}_{1,1}, \mathfrak{v}_{1,2}, \mathfrak{v}_{2,1}, \mathfrak{v}_{2,2})$ is a Diffie-Hellman tuple.

Given a Square Diffie-Hellman tuple $(\tau_1 = h, \tau_2 = h^{\tilde{\tau}}, \tau_3 = h^{\tilde{\tau}^2}) \in \mathbb{G}_1^3$, one first commits $\tilde{\tau}$: $\mathsf{Com} = (\mathfrak{c} = \mathfrak{v}_{2,1}^{\tilde{\tau}} \mathfrak{v}_{1,1}^{\mu}, \mathfrak{d} = \mathfrak{v}_{2,2}^{\tilde{\tau}} \mathfrak{g}^{\tilde{\tau}} \mathfrak{v}_{1,2}^{\mu})$, for a random $\mu \xleftarrow{\$} \mathbb{Z}_p$, and one sets $\pi_1 = \tau_1^{\mu}$ and $\pi_2 = \tau_2^{\mu}$, which satisfy

$$e(\tau_1, \mathfrak{c}) = e(\tau_2, \mathfrak{v}_{2,1}) \cdot e(\pi_1, \mathfrak{v}_{1,1}) \qquad e(\tau_1, \mathfrak{d}) = e(\tau_2, \mathfrak{v}_{2,2} \cdot \mathfrak{g}) \cdot e(\pi_1, \mathfrak{v}_{1,2})$$
$$e(\tau_2, \mathfrak{c}) = e(\tau_3, \mathfrak{v}_{2,1}) \cdot e(\pi_2, \mathfrak{v}_{1,1}) \qquad e(\tau_2, \mathfrak{d}) = e(\tau_3, \mathfrak{v}_{2,2} \cdot \mathfrak{g}) \cdot e(\pi_2, \mathfrak{v}_{1,2})$$

The proof $\mathsf{proof} = (\mathfrak{c}, \mathfrak{d}, \pi_1, \pi_2)$, when it satisfies the above relations, guarantees that (τ_1, τ_2, τ_3) is a Square Diffie-Hellman tuple. This proof is furthermore zero-knowledge, under the DDH assumption in $\mathbb{G}_2$: by switching $(\mathfrak{v}_{1,1}, \mathfrak{v}_{1,2}, \mathfrak{v}_{2,1}, \mathfrak{g} \times \mathfrak{v}_{2,2})$ into a Diffie-Hellman tuple, one can simulate the proof, as the commitment is perfectly hiding.

As explained in [HPP19], one can apply a batch verification [BFI+10], and pack them in a unique one with random scalars $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \xleftarrow{\$} \mathbb{Z}_p$:

$$e(\tau_1^{x_{2,1}} \tau_2^{x_{2,2}}, \mathfrak{c}^{x_{1,1}} \mathfrak{d}^{x_{1,2}}) = e(\tau_2^{x_{2,1}} \tau_3^{x_{2,2}}, \mathfrak{v}_{2,1}^{x_{1,1}} \mathfrak{v}_{2,2}^{x_{1,2}} \mathfrak{g}^{x_{1,2}}) \times e(\pi_1^{x_{2,1}} \pi_2^{x_{2,2}}, \mathfrak{v}_{1,1}^{x_{1,1}} \mathfrak{v}_{1,2}^{x_{1,2}})$$

One thus just has to compute 13 exponentiations and 3 pairing evaluations for the verification, instead of 12 pairing evaluations.

F.3 Groth-Sahai Proof for Square Diffie-Hellman Tracing

For the proof of tracing, one wants to show $\tau' \sim \tau$, where τ is the reference tag for a user (certified at the registration time). With the tracing key $\mathsf{utk} = \mathfrak{g}^{\tilde{\tau}}$, one needs to show

$$e(\tau_1, \mathsf{utk}) = e(\tau_2, \mathfrak{g}) \qquad e(\tau_2, \mathsf{utk}) = e(\tau_3, \mathfrak{g})$$
$$e(\tau'_1, \mathsf{utk}) = e(\tau'_2, \mathfrak{g}) \qquad e(\tau'_2, \mathsf{utk}) = e(\tau'_3, \mathfrak{g})$$

but without revealing $\mathsf{utk} \in \mathbb{G}_2$. This is equivalent, for random $\alpha_1, \alpha_2, \alpha'_1, \alpha'_2 \xleftarrow{\$} \mathbb{Z}_p$, to have:

$$e(T_1, \mathsf{utk}) = e(T_2, \mathfrak{g}) \qquad \text{with} \qquad T_1 = {\tau_1}^{\alpha_1} \cdot {\tau_2}^{\alpha_2} \cdot {\tau'_1}^{\alpha'_1} \cdot {\tau'_2}^{\alpha'_2}$$
$$T_2 = {\tau_2}^{\alpha_1} \cdot {\tau_3}^{\alpha_2} \cdot {\tau'_2}^{\alpha'_1} \cdot {\tau'_2}^{\alpha'_2}$$

One can commit utk: as above, with the reference string $(\mathfrak{v}_{1,1}, \mathfrak{v}_{1,2}, \mathfrak{v}_{2,1}, \mathfrak{v}_{2,2}) \in \mathbb{G}_2^4$, such that $(\mathfrak{v}_{1,1}, \mathfrak{v}_{1,2}, \mathfrak{v}_{2,1}, \mathfrak{v}_{2,2})$ is a Diffie-Hellman tuple, one computes $\mathsf{Com} = (\mathfrak{c} = \mathfrak{v}_{2,1}^{\lambda}\mathfrak{v}_{1,1}^{\mu}, \mathfrak{d} = \mathfrak{v}_{2,2}^{\lambda}\mathfrak{v}_{1,2}^{\mu} \times \mathsf{utk})$, for random $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$, and one sets $\pi_1 = T_1^{\lambda}$ and $\pi_2 = T_1^{\mu}$, which should satisfy

$$e(T_1, \mathfrak{c}) = e(\pi_1, \mathfrak{v}_{2,1}) \cdot e(\pi_2, \mathfrak{v}_{1,1}) \qquad e(T_1, \mathfrak{d}) = e(T_2, \mathfrak{g}) \cdot e(\pi_1, \mathfrak{v}_{2,2}) \cdot e(\pi_2, \mathfrak{v}_{1,2})$$

The random values $\alpha_1, \alpha_2, \alpha'_1, \alpha'_2$ can be either chosen by the verifier in case of interactive proof, or set from $H(\tau_1, \tau_2, \tau_3, \tau'_1, \tau'_2, \tau'_3)$.