Anonymous Attestations Made Practical

Amira Barki*
Trusted Labs
France
amira.barki@trusted-labs.com

Nicolas Desmoulins
Orange Labs
France
nicolas.desmoulins@orange.com

Saïd Gharout
Orange Labs
France
said.gharout@orange.com

Jacques Traoré
Orange Labs
France
jacques.traore@orange.com

ABSTRACT

Direct Anonymous Attestation (DAA) is a privacy preserving authentication protocol initially designed for Trusted Platform Modules (TPMs). This cryptographic protocol, and some of its extensions such as Intel's Enhanced Privacy ID (EPID), have been widely deployed in millions of chips. Usually, part of the attestation computation is delegated to the host (in most cases, either a PC or a smartphone) embedding the TPM, which is generally much more powerful. However, in Machine-to-Machine (M2M) and Internet of Things (IoT) use cases, the host may be as resource constrained as the TPM. Furthermore, any malware residing in the host may enable the tracking of the TPM owner.

In this paper, we propose an efficient DAA scheme, defined on elliptic curves, that involves bilinear pairings computations only on the verifier's side. Consequently, all computations on the platform side required to verify the validity of a group signing key or to generate a DAA can be, contrarily to previous solutions, entirely carried out by a resource constrained TPM. Our DAA scheme, which is more efficient than all existing DAA schemes, is formally proven secure under a variant of the LRSW assumption and can be extended to support private key and signature based revocations as well as group signing keys with attributes. As it is suitable for resource constrained environments such as SIM cards, our DAA scheme can be of particular interest for M2M applications involving a SIM card. More precisely, we show how to design a privacy-preserving authentication protocol for embedded SIMs (eSIM) so as to cope with a real issue that has arisen at GSM Association (GSMA). By implementing our DAA scheme on a Global Platform compliant SIM card, we show its efficiency and suitability for real-world use cases. Actually, a TPM can be anonymously authenticated in only 169 ms.

*Work done while being at Orange Labs

CCS CONCEPTS

•Security and privacy → Digital signatures; Privacy-preserving protocols;

KEYWORDS

DAA, EPID, authentication, privacy, SIM cards.

ACM Reference format:
Amira Barki, Nicolas Desmoulins, Saïd Gharout, and Jacques Traoré. 2017. Anonymous Attestations Made Practical. In *Proceedings of WiSec '17 , Boston, MA, USA, July 18-20, 2017,* 12 pages.
DOI: 10.1145/3098243.3098258

1 INTRODUCTION

Direct Anonymous Attestations (DAAs), introduced by Brickell, Camenisch and Chen [7] and standardized by the Trusted Computing Group (TCG) in 2004 [34], originally aimed to allow the anonymous authentication of a particular hardware module, known as Trusted Platform Module (TPM) (*e.g.* a secure element in a smartphone), to an external party (*e.g.* a service provider). Thereby, the TPM can be remotely authenticated whilst preserving the privacy of its owner. Over the last years, DAAs have been widely deployed. Indeed, more than 500 million TPMs have been sold and Intel's DAA extension, known as EPID, have been in use in over 2.4 billion devices [22].

DAA schemes can be considered as a particular group signature scheme that does not support signature opening capabilities, but instead, ensures the linkability of signatures created by the same module (*i.e.* using the same key) with respect to the same *basename*, denoted bsn. Owing to the limited computational capabilities of TPMs, the signature is jointly created by the TPM and the host embedding it (*e.g.* a PC or a smartphone), which is more powerful but untrustworthy.

Related work. Following Brickell *et al.* scheme [7], several DAA schemes have been proposed [7, 9, 11–18, 21], with the main purpose of improving efficiency, reducing the required TPM resources or providing formal security definitions. Some of these proposals [7, 9, 14] have even been adopted by the International Organization for Standardization (ISO) [27].

A number of DAA extensions have also been proposed. Brickell *et al.* [8, 10] introduced a DAA scheme, referred to as Enhanced Privacy ID (EPID), which features enhanced revocation capabilities. Indeed, EPID enables the revocation of a platform based on a signature that it has previously generated, without knowledge of the platform secret key. Recently, Chen *et al.* [17] proposed DAA with attributes (DAA-A) which, as its name implies, support group signing keys with attributes that can be selectively disclosed. For instance, when producing the signature, the TPM can reveal the attribute corresponding to its group signing key expiry date whilst keeping the remaining ones hidden.

In [4], Bernhard *et al.* introduced a special DAA scheme, referred to as pre-DAA, where all the computations on the platform side are performed by the TPM and none is delegated to the host. However, Bernhard *et al.* scheme requires pairing computations during the *join* phase, which makes it unsuitable for resource constrained environments such as SIM cards.

Furthermore, some works [2, 20, 28] have focused on the implementation of DAA or anonymous credentials schemes in resource constrained environment such as smart cards.

Unfortunately, many DAA schemes turned out to be insecure. Indeed, Camenisch *et al.* [13] have recently shown that some DAA security models do not really cover all the required security properties. Worse still, they identified flaws in many LRSW-based DAA schemes [13] as well as in the security proofs of several DAA schemes based on the q-Strong Diffie Hellman ($q-$ SDH) assumption [11].

Contributions. In this paper, our contributions are twofold. First, we propose an efficient LRSW-based pre-DAA scheme where, contrarily to previous proposals, all computations on the platform side are carried out by the TPM. This avoids any tracking of the user through malwares that may reside in the host. Since it is pairing-free on the platform side, our proposal is suitable for resource constrained environments like SIM cards (which cannot handle pairing computations or even computations in either $\mathbb{G}_2$ or $\mathbb{G}_T$). Furthermore, it can be easily extended to support private key and signature based revocations as well as group signing keys with attributes. Our pre-DAA scheme is proven to ensure all the required security properties in the game-based security model defined by Bernhard *et al* [4] (which models the host and the TPM as a same entity). By implementing it on a standard SIM card, we confirm its efficiency and suitability for resource constrained environments and concrete real world applications.
Then, based on our pre-DAA scheme, we address an issue that has arisen at GSMA by designing a privacy-preserving protocol enabling the anonymous authentication and identification of an embedded SIM (eSIM) to a Mobile Network Operator (MNO) *independent* entity known as discovery server. Thereby, following a change of MNO, an eSIM can be remotely provisioned with its new network profile while protecting the privacy of its owner against a malicious discovery server.

Organization. The paper is structured as follows. Section 2 introduces the necessary building blocks. Then, Section 3 provides an overview on DAA schemes as well as their security requirements, and details our construction. In Section 4, we briefly recall the eSIM remote profile provisioning procedure before describing our privacy-preserving protocol. Finally, Section 5 provides efficiency evaluations as well as implementation benchmarks of our pre-DAA scheme on a GlobalPlatform compliant SIM card.

2 BUILDINGS BLOCKS

In this section, we introduce the main building blocks of our pre-DAA scheme. First, we review bilinear maps and zero-knowledge proofs of knowledge (ZKPKs). Then, we define the required computational assumptions.

In the sequel, we use the notation $x \overset{R}{\longleftarrow} X$ or $x \in_R X$ to state that x is chosen uniformly at random from the set X.

2.1 Bilinear maps

Bilinear groups are a set of three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a bilinear map, also called pairing, $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ satisfying the following three properties:

- For all $g \in \mathbb{G}_1, \tilde{g} \in \mathbb{G}_2$ and $(a, b) \in \mathbb{Z}_p^2$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{a.b}$;
- For $g \neq 1_{\mathbb{G}_1}$ and $\tilde{g} \neq 1_{\mathbb{G}_2}$, $e(g, \tilde{g}) \neq 1_{\mathbb{G}_T}$;
- e is efficiently computable.

Pairings are classified into three main types depending on the existence (or not) of computable isomorphism(s) between $\mathbb{G}_1$ and $\mathbb{G}_2$. It is noteworthy to mention that Type-3 pairings (*i.e* pairing groups where there exists no efficient isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$) provide the most efficient operations in $\mathbb{G}_1$ [1].

2.2 Zero-Knowledge Proof of Knowledge

Zero-Knowledge Proofs of Knowledge (ZKPKs) allow a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that he knows some secrets verifying a given statement without revealing anything else about them. In the sequel, they are denoted by the usual notation $\pi = \text{PoK}\{\alpha, \beta : \textit{statements about } \alpha, \beta\}$ where Greek letters correspond to the knowledge of $\mathcal{P}$.

A ZKPK should satisfy three properties, namely (1) *completeness* (*i.e.* a valid prover should be able to convince an honest verifier with overwhelming probability), (2) *soundness* (*i.e.* a malicious prover should be rejected with overwhelming probability), and (3) *zero-knowledge* (*i.e.* the proofs reveal no information about the secret(s) except that they satisfy the given statement).

In addition to classical ZKPKs (such as proofs of knowledge of a discrete logarithm or proofs of equality of discrete logarithms), our pre-DAA scheme relies on ZKPK of inequality of discrete logarithms (for signature-based revocations). Such proofs have been introduced by Brands in [5].

2.3 Computational hardness assumptions

The security of our DAA schemes relies on the by now classical following assumptions.

2.3.1 LRSW Assumption. Let $\mathbb{G}$ be a cyclic group of prime order p and g be a generator of $\mathbb{G}$. The LRSW assumption [29] states that, given $X_0 = g^{x_0}$, $X_1 = g^{x_1}$ and an oracle $\mathcal{O}^{\text{LRSW}}_{X_0,X_1}$ that takes on input a message $m \in \mathbb{Z}_p$ and outputs $A = (a, a^{x_1}, a^{x_0+mx_0x_1})$ such that a is randomly chosen, it is hard to compute $A' = (a', a'^{x_1}, a'^{x_0+mx_0x_1})$ for another value a', if m has not already been queried to the oracle $\mathcal{O}^{\text{LRSW}}_{X_0,X_1}$.

To prove the security of their signature schemes, Pointcheval and Sanders [32] introduced two new assumptions which are variants of the classical LRSW assumption in Type-3 bilinear groups. In what follows, we recall the first assumption which is proven secure in the generic group model [30, 31, 33].

2.3.2 Assumption 1. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$ be a bilinear group setting of Type 3, with h and $\tilde{h}$ two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, which are of prime order p. The assumption 1 [32] states that, given $(h, X_1, \tilde{h}, \tilde{X}_0, \tilde{X}_1)$ where $X_1 = h^{x_1}$, $\tilde{X}_0 = \tilde{h}^{x_0}$ and $\tilde{X}_1 = \tilde{h}^{x_1}$ such that $x_0, x_1 \in_R \mathbb{Z}_p^*$ and having an unlimited access to an oracle O_1 which, on input $m \in \mathbb{Z}_p$, chooses a random $u \in \mathbb{G}_1$ and outputs the pair $P = (u, u^{x_0+mx_1})$, it is hard to efficiently generate a valid pair P, with $u \neq 1_{\mathbb{G}_1}$, for a new m^* that has not been queried to O_1.

*2.3.3 Decisional Diffie-Hellman (*DDH*) assumption.* Let $\mathbb{G}$ be a cyclic group of prime order p. The Decisional Diffie-Hellman DDH assumption [6] states that, given a generator $g \in \mathbb{G}$, two random elements $g^a, g^b \in \mathbb{G}$ and a candidate $X \in \mathbb{G}$, it is hard to decide whether $X = g^{ab}$ or not.

*2.3.4 One-More Discrete Logarithm (*OMDL*) assumption.* Let $\mathbb{G}$ be a cyclic group of prime order p, g a generator of $\mathbb{G}$, O_1 a challenge oracle that returns a random element $Y \in \mathbb{G}$ and O_2 a discrete logarithm oracle with respect to the base g. The One-More Discrete Logarithm OMDL assumption [3] states that, after t queries to O_1 (where t is chosen by the adversary) and **at most** $(t-1)$ queries to O_2, it is hard to recover the discrete logarithms of all received t elements.

*2.3.5 eXternal Diffie-Hellman (*XDH*) Assumption.* Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The eXternal Diffie-Hellman assumption, denoted by XDH, states that the DDH assumption holds in $\mathbb{G}_1$.

3 A PRACTICAL AND SECURE PRE-DAA

3.1 Overview

Stakeholders. Usually, a DAA scheme involves four main entities: an issuer $\mathcal{I}$, a TPM $\mathcal{P}$, a host $\mathcal{H}$ and a verifier $\mathcal{V}$. In our pre-DAA scheme, all computations on the platform side are carried out by the TPM. Thus, it does not involve any host. In the sequel, the term *platform* will sometimes be used to refer to the TPM.

Overview. Our pre-DAA scheme consists of five phases: (1) A *Setup* phase where the system public parameters are initialized. (2) A *Key Generation* phase enables the generation of the issuer's private and public keys. (3) The *Join* phase allows a platform $\mathcal{P}$ to obtain its group signing key, which is issued by $\mathcal{I}$. (4) During the *Sign* phase, the platform $\mathcal{P}$ uses its group signing key to anonymously sign a message m with respect to a given basename bsn. Finally, (5) the *Verify* phase enables to check the validity of a signature.

3.2 Security requirements

In addition to the usual *correctness* property, a pre-DAA scheme should satisfy three security properties, namely *traceability, non-frameability* and *anonymity* [4]. Roughly speaking, they are defined as follows (formal definitions are provided in Appendix A):

- *Traceability*: no adversary should be able to generate: (1) a valid signature which cannot be traced to a secret key that has been committed to during an execution of the Join protocol; or (2) two *unlinkable* signatures with respect to the same basename and under the same secret key.
- *Non-frameability*: no adversary should be able to output: (1) a signature that can be traced to a given user i who has not produced a signature on the corresponding message/basename pair; or (2) two signatures from different signers that are linkable even though they should not.
- *Anonymity*: given two identities and a signature, an adversary should not be able to decide which one generated the signature.

3.3 Our pre-DAA scheme

In this section, we design an efficient and secure LRSW-based pre-DAA scheme which, unlike previous proposals, requires no pairing computations on the platform side to verify the validity of a group signing key or to generate a DAA. This makes it suitable for resource constrained environments such as SIM cards (which cannot handle pairing computations). Besides, since all computations are performed by the TPM, our pre-DAA scheme avoids the tracking of the user through malwares that may reside in the host.

Hereinafter, we detail the five phases of our pre-DAA scheme and depict the three main ones (*i.e.* Join, Sign and Verify) in Figure 1.

3.3.1 Setup. Generate the system public parameters denoted by $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, h, \tilde{h}, \mathcal{H}, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups of order p, a k-bit prime, g, h are two random generators of $\mathbb{G}_1$ whereas $\tilde{h}$ is a random generator of $\mathbb{G}_2$, $\mathcal{H}$ is a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}_1$ (that will be considered as a random oracle in the security proof) and $e : \mathbb{G}_1 \text{ x } \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a Type-3 bilinear map.

3.3.2 Key Generation. Choose two random values $x_0, x_1 \in_R \mathbb{Z}_p^*$ as the issuer's private key, denoted *gmsk*, and compute the corresponding public key $gmpk = (C_{x_0} = g^{x_0} h^{\tilde{x_0}}, X_1 = h^{x_1}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1})$ where $\tilde{x_0} \in_R \mathbb{Z}_p^*$. Then, prove knowledge of the corresponding private keys x_0, x_1 and $\tilde{x_0}$ by providing a ZKPK π defined as $\pi = \{\alpha, \beta, \gamma : C_{x_0} = g^\alpha h^\beta \wedge X_1 = h^\gamma \wedge \tilde{X}_0 = \tilde{h}^\alpha \wedge \tilde{X}_1 = \tilde{h}^\gamma\}$.

Each platform $\mathcal{P}$ is provided with a public/private key pair (esk, epk) which is used to authenticate it during the Join phase (*i.e.* before receiving its group signing key). Each platform also selects a random value $s_1 \in_R \mathbb{Z}_p^*$ as its secret key sk.

3.3.3 Join. To get its group signing key, the platform $\mathcal{P}$ and the issuer $\mathcal{I}$ engage in the following protocol. First, the platform computes $C_{s_1} = X_1^{s_1}$, a commitment to its secret s_1. The platform also builds a ZKPK π_1 defined as $\pi_1 = \text{PoK}\{\alpha : C_{s_1} = X_1^\alpha\}$ and computes a signature $S = Sign_{esk}(C_{s_1})$ on it computed using esk. Once done, it provides the issuer with C_{s_1}, S and the proof π_1. Upon their receipt, the issuer $\mathcal{I}$ first verifies that $C_{s_1} \neq 1$ and checks the validity of the signature and proof π_1. Then, he picks $b \in_R \mathbb{Z}_p^*$ and generates a pair (u, u') associated to s_1 where $u = h^b$ and $u' = u^{x_0} C_{s_1}^b = u^{x_0+s_1x_1}$. $\mathcal{I}$ also builds a ZKPK π_2 to ensure that the pair (u, u') is well-formed. The proof π_2 is defined as $\pi_2 = \text{PoK}\{\alpha, \beta, \gamma : u = h^\alpha \wedge u' = u^\beta C_{s_1}^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}$. Finally, the platform is provided with (u, u') along with π_2. If $u \neq 1$ and the proof π_2 is valid, the platform sets its group signing key as $gsk = (u, u')$.

	Public Input: pp, epk, $gmpk = (C_{x_0}, X_1, \tilde{X}_0, \tilde{X}_1)$	
	(1) Join (*i.e.* Issuance of a group signing key)	
Platform $\mathcal{P}$		**Issuer $\mathcal{I}$**
		Private Input: x_0, x_1, x_2
Compute $C_{s_1} \leftarrow X_1^{s_1}, S = Sign_{esk}(C_{s_1})$		
Build $\pi_1 = \mathrm{PoK}\{\alpha : C_{s_1} = X_1^{\alpha}\}$	$\xrightarrow{C_{s_1}, S, \pi_1}$	**Check** π_1 and S
		Choose $b \overset{R}{\leftarrow} \mathbb{Z}_p^*$
		Compute $u \leftarrow h^b$, $u' \leftarrow u^{x_0} C_{s_1}^b$
Check π_2	$\xleftarrow{u, u', \pi_2}$	**Build** $\pi_2 = \mathrm{PoK}\{\alpha, \beta, \gamma : u = h^{\alpha} \wedge u' = u^{\beta} C_{s_1}^{\alpha}$
$gsk \leftarrow (u, u')$ where $u' = u^{x_0 + s_1 x_1}$		$\wedge\, C_{x_0} = g^{\beta} h^{\gamma}\}$
	(2) Sign and Verify Protocols	
Platform $\mathcal{P}$		**Verifier $\mathcal{V}$**
	Additional Public Input: bsn	
Private Input: (u, u'), s_1		
Choose $l \overset{R}{\leftarrow} \mathbb{Z}_p^*$		
Compute $w \leftarrow u^l$, $w' = (u')^l$		
$c_1 \leftarrow w^{s_1}$, $T \leftarrow \mathcal{H}(\mathsf{bsn})^{s_1}$		
Build $\pi_3 = \mathrm{PoK}\{\alpha : c_1 = w^{\alpha} \wedge T = \mathcal{H}(\mathsf{bsn})^{\alpha}\}$	$\xrightarrow{m, w, w', c_1, T, \pi_3}$	**Check** if $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \overset{?}{=} e(w', \tilde{h})$
		Check π_3
Details of the proof π_3:		
Choose $a_1 \overset{R}{\leftarrow} \mathbb{Z}_p^*$		
Compute $t_1 \leftarrow w^{a_1}$, $t_2 \leftarrow \mathcal{H}(\mathsf{bsn})^{a_1}$		
Compute $c = \mathcal{H}(t_1, t_2, m, Ch)$	$\xleftarrow{Ch}$	**Choose** $Ch \in_R \mathbb{Z}_p^*$
$R_1 \leftarrow a_1 + cs_1 (mod\ p)$	$\xrightarrow{c, R_1}$	**Compute** $t_1' = w^{R_1} c_1^{-c}$
		$t_2' = \mathcal{H}(\mathsf{bsn})^{R_1} T^{-c}$
		Check if $\mathcal{H}(t_1', t_2', m, Ch) \overset{?}{=} c$

Figure 1: Our LRSW-based pre-DAA scheme

3.3.4 Sign. To anonymously sign a message $m \in \{0,1\}^*$ with respect to the basename bsn, the platform $\mathcal{P}$ proceeds as follows. First, it randomly selects $l \in_R \mathbb{Z}_p^*$, then computes $w = u^l$ and $w' = (u')^l$, a randomized version of its group signing key. It also computes $c_1 = w^{s_1}$ and $T = \mathcal{H}(\mathsf{bsn})^{s_1}$. Moreover, $\mathcal{P}$ needs to prove that the same secret is committed to in both c_1 and T.

Accordingly, $\mathcal{P}$ provides $\mathcal{V}$ with the message m as well as the signature $\sigma = (w, w', c_1, T, \pi_3)$ where π_3 is a ZKPK defined as $\pi_3 = \mathrm{PoK}\{\alpha : c_1 = w^{\alpha} \wedge T = \mathcal{H}(\mathsf{bsn})^{\alpha}\}$ (see Figure 1).

3.3.5 Verify. To check the validity of a signature σ defined as $\sigma = (w, w', c_1, T, \pi_3)$, $\mathcal{V}$ first verifies that $w \neq 1, T \neq 1$ and $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) = e(w', \tilde{h})$. If so, it checks the validity of π_3. The signature is considered valid if, and only if, all checks succeed.

Theorem 3.1. *Our pre-DAA scheme provides* traceability *under the assumption* 1, non-frameability *under the* OMDL *assumption and* anonymity *under the* XDH *assumption, in the random oracle model*[1].

3.4 Possible extensions

In this section, we explain the possible extensions of our pre-DAA scheme so as to support private-key and signature-based revocations as well as group signing keys with attributes.

3.4.1 Platform Revocation. Our scheme can be extended to support private-key as well as signature-based revocations. To this end, we introduce an additional entity: a revocation manager $\mathcal{R}$ (it could be the same entity as the issuer). $\mathcal{R}$ handles the management of two revocations lists: Key_{RL}, a private key based revocation list, and Sign_{RL}, a signature based revocation list. Initially, both lists are empty.

- *Revocation based on a private key*: to revoke the platform whose secret key and associated group signing key are respectively s_1^i and (u, u'), $\mathcal{R}$ first verifies the correctness of (u, u') with respect to s_1^i. To do so, it checks that $u \neq 1$ and $e(u, \tilde{X}_0) \cdot e(u, \tilde{X}_1)^{s_1^i} = e(u', \tilde{h})$. If both checks succeed, $\mathcal{R}$ updates the private key based revocation list by appending s_1^i to it: $\mathsf{Key}_{RL} = \mathsf{Key}_{RL} \cup \{s_1^i\}$. Henceforth, upon the receipt of a signature $\sigma = (w, w', c_1, T, \pi_3)$, the verifier $\mathcal{V}$ additionally computes $T_j = \mathcal{H}(\mathsf{bsn})^{s_1^j}, \forall s_1^j \in \mathsf{Key}_{RL}$ and checks that none is equal to T (*i.e.* $T \neq T_j, \forall j$). If so, the signature is considered valid (*i.e.* it has not been produced by a revoked platform).
- *Revocation based on signature*: to revoke the platform which has created the signature σ_i on the message m, $\mathcal{R}$ first verifies the validity of the signature σ_i. To do so, it checks that $w \neq 1$, $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \overset{?}{=} e(w', \tilde{h})$ and π_3 is valid. If so, $\mathcal{R}$ updates the signature based revocation list $\mathsf{Sign}_{RL} = \mathsf{Sign}_{RL} \cup \{(\mathsf{bsn_i}, T_i = \mathcal{H}(\mathsf{bsn_i})^{s_1^i})\}$. To support this type

[1]The proofs are provided in Appendix B.

of revocation, some changes are required in the *Sign* protocol. More precisely, when signing a message m, the platform $\mathcal{P}$ must also build a ZKPK π_j^r associated to each pair $(\mathcal{H}(\mathsf{bsn_j}), T_j = \mathcal{H}(\mathsf{bsn_j})^{s_1^j}) \in \mathtt{Sign}_{RL}$. The proof is defined as $\pi_j^r = \mathrm{PoK}\{\alpha : T = \mathcal{H}(\mathsf{bsn})^\alpha \wedge T_j \neq \mathcal{H}(\mathsf{bsn}_j)^\alpha\}$[2].

3.4.2 *From pre-DAA to pre-DAA-A.* Our pre-DAA scheme can also be extended to support group signing keys with one or several attributes $t_1, \ldots, t_n$. To do so, some changes are required and the issuer should have as many additional private keys as attributes. Hereinafter, we describe the extension of our pre-DAA in the case of one attribute denoted s_2. Since this attribute may have a small entropy (e.g. an expiry date), unlike s_1, we provide a pre-DAA-A scheme which perfectly hides its value.

The *Setup* phase remains the same whereas the other phases are updated as follows.

Key Generation. The issuer holds an additional secret key x_2 which is chosen at random and associated to the public values $X_2 = h^{x_2}$ and $\tilde{X}_2 = \tilde{h}^{x_2}$. It should also prove knowledge of the corresponding private keys x_0, $\tilde{x_0}$, x_1 and x_2 by providing a ZKPK π defined as $\pi = \{\alpha, \beta, \gamma, \delta : C_{x_0} = g^\alpha h^\beta \wedge \tilde{X}_0 = \tilde{h}^\alpha \wedge X_1 = h^\gamma \wedge \tilde{X}_1 = \tilde{h}^\gamma \wedge X_2 = h^\delta \wedge \tilde{X}_2 = \tilde{h}^\delta\}$.

Join. To get its group signing key on the secret $s_1 \neq 0$ and the attribute s_2, $\mathcal{P}$ computes $C = X_1^{s_1} X_2^{s_2}$, a commitment to s_1 and s_2. The TPM also builds a ZKPK π_1[3] defined as $\pi_1 = \mathrm{PoK}\{\alpha, \beta : C = X_1^\alpha X_2^\beta\}$. Once done, it provides the issuer with C as well as the proof π_1 and a signature $S = Sign_{esk}(C)$ on it computed using esk. Upon their receipt, the issuer first verifies the validity of the signature and the proof π_1. Then, he picks $b \in_R \mathbb{Z}_p^*$ and generates a pair (u, u') associated to both s_1 and s_2 where $u = h^b$ and $u' = u^{x_0} C^b$. Finally, the user is provided with (u, u') along with a ZKPK π_2 defined as $\pi_2 = \mathrm{PoK}\{\alpha, \beta, \gamma : u = h^\alpha \wedge u' = u^\beta C^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}$. If $u \neq 1$ and the proof π_2 is valid, the platform sets its group signing key as $gsk = (u, u') = (u, u^{x_0 + s_1 x_1 + s_2 x_2})$.

Sign. $\mathcal{P}$ first computes w, w' and T as previously. Then, it selects $z_1, z_2, z_3 \in_R \mathbb{Z}_p^*$ at random and computes $c_1 = w^{s_1} h^{z_1}, c_2 = w^{s_2} h^{z_2}, c' = w' g^{z_3}, V = g^{-z_3} X_1^{z_1} X_2^{z_2}$. A signature on m and for bsn consists of $\sigma = (w, c_1, c_2, c', V, T, \pi_3)$ where $\pi_3 = \mathrm{PoK}\{\alpha, \beta, \gamma, \lambda, \phi : c_1 = w^\alpha h^\gamma \wedge c_2 = w^\beta h^\lambda \wedge V = g^{-\phi} X_1^\gamma X_2^\lambda \wedge T = \mathcal{H}(\mathsf{bsn})^\alpha\}$.

Verify. To check the validity of a signature σ with respect to a message m and a basename bsn, $\mathcal{V}$ first verifies that $w \neq 1$ and $T \neq 1$, then checks that $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \cdot e(c_2, \tilde{X}_2) = e(c'V, \tilde{h})$ and π_3 is valid.

Hereinafter, we show that if all checks succeed, then this proves that $\mathcal{P}$ knows a valid group signing key (w, w') on a secret s_1 and an attribute s_2 such that $T = \mathcal{H}(\mathsf{bsn})^{s_1}$. Indeed, owing to the soundness property of π_3, $\mathcal{P}$ knows $\alpha, \beta, \gamma, \lambda$ and ϕ such that $c_1 = w^\alpha h^\gamma$, $c_2 = w^\beta h^\lambda$, $V = g^{-\phi} X_1^\gamma X_2^\lambda$ and $T = \mathcal{H}(\mathsf{bsn})^\alpha$. Since we know that $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \cdot e(c_2, \tilde{X}_2) = e(c'V, \tilde{h})$, we therefore have:

$$e(w^{x_0} w^{\alpha x_1} h^{\gamma x_1} w^{\beta x_2} h^{\lambda x_2}, \tilde{h}) = e(c' g^{-\phi} h^{\gamma x_1} h^{\lambda x_2}, \tilde{h})$$

This implies that

$$w^{x_0 + \alpha x_1 + \beta x_2} h^{\gamma x_1} h^{\lambda x_2} = c' g^{-\phi} h^{\gamma x_1} h^{\lambda x_2}$$

and thus

$$w^{x_0 + \alpha x_1 + \beta x_2} = c' g^{-\phi}$$

Let us denote by $s_1 = \alpha, s_2 = \beta$ and $w' = c' g^{-\phi}$. We can therefore conclude that $\mathcal{P}$ knows a valid group signing key (w, w') on a secret s_1 and an attribute s_2 such that $T = \mathcal{H}(\mathsf{bsn})^{s_1}$.

4 EMBEDDED SIM USE CASE

4.1 Problem statement

Some Machine to Machine (M2M) devices, like smart meters for example, are not designed in a way enabling the removal or replacement of the SIM card. In fact, the latter is either hardly accessible or even welded in the device at its manufacture. Thus, there is no way to switch from one network operator to another.

To cope with this issue and provide more flexibility, the GSM Association (GSMA) introduced a new generation of SIM cards referred to as embedded SIM (eSIM), or eUICC, which enables the change of the subscription associated to a SIM card while keeping the same chip and level of security as classical SIM cards. The GSMA also proposed an architecture for Over-The-Air (OTA) remote SIM provisioning with its new network profile [23–26]. This architecture involves a Mobile Network Operator (MNO) *independent* third party known as Subscription Manager Discovery Server (SM-DS). The latter mainly aims to provide an eSIM with the address of the server, called Subscription Manager-Data Preparation+ (SM-DP+), that has generated its new profile. Thereby, the eSIM can get in touch with it to download its new network profile.

However, if no privacy-preserving mechanisms are set up, the SM-DS would be able to trace the eSIM subscription changes all along its life cycle, hence entailing some privacy concerns.

In this subsection, we briefly explain how our pre-DAA scheme can be used to design a privacy-preserving protocol that enables the discovery server (SM-DS) to anonymously authenticate and identify the relevant eSIM. Thereby, it can provide the eSIM with the address of the corresponding SM-DP+ without knowing with which eSIM it is currently interacting (*i.e.* without knowing the unique identifier, denoted by EID, of the eSIM).

4.2 Overview of the eSIM remote profile provisioning procedure

The remote eSIM provisioning architecture involves four main stakeholders, namely an embedded SIM (eSIM) which is uniquely identified by its *eUICC identifier* EID, a mobile network operator (MNO), a Subscription Manager-Data Preparation (SM-DP+) and an MNO *independent* third party known as a Subscription Manager-Discovery Server (SM-DS). It is noteworthy to mention that the eUICC Manufacturer (EUM) is also indirectly involved as it handles the issuance of the group signing key that enables the authentication of the eSIM to the SM-DS and SM-DP+.

The main steps that take place whenever the user changes the subscription associated to an eSIM are as follows (see Figure 2):

[2]Such a proof of inequality of discrete logarithm has been introduced in [5].
[3]For concurrent security, we could use the Damgard protocol [19] which converts any Σ protocol into a three-round interactive ZKPK secure under concurrent composition.

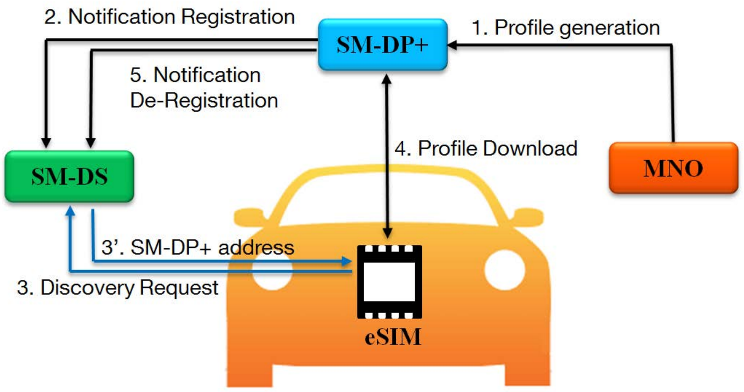

Figure 2: Remote Profile Provisioning Procedure (Automotive use case)

(1) *Profile Generation*: The MNO asks a specific SM-DP+ to generate the eSIM new profile. The latter will be securely stored until its download by the corresponding eSIM.
(2) *Notification Registration*: Once the profile has been generated, the SM-DP+ sends a notification registration to the SM-DS. It aims to inform the SM-DS of the availability of a new profile for the eSIM identified by EID in a given SM-DP+.
(3) *Discovery Request*: The eSIM queries its default SM-DS to know whether there is a pending new profile for it. If so, after authenticating and identifying the eSIM, the SM-DS provides it with the address of the SM-DP+ that has generated its new profile.
(4) *Profile Download*: The eSIM contacts the relevant SM-DP+ so as to download its new profile. After mutually authenticating each other, the eSIM is finally securely provided with its new network profile.
(5) *Notification De-Registration*: Following a successful profile download, the SM-DP+ notifies the SM-DS. Thus, the latter can delete the corresponding notification as it is no longer useful.

4.3 A privacy-preserving protocol for eSIM

Hereinafter, we explain how to design a privacy-preserving authentication and identification protocol for eSIM using our pre-DAA scheme as the main building block. Thereby, an eSIM can recover the address of the relevant SM-DP+ whilst preserving its owner's privacy with respect to the MNO-independent SM-DS.

Our privacy-preserving authentication and identification protocol for eSIM consists of the following five phases:

Setup. It is exactly the same as the *Setup* phase of our pre-DAA scheme.

Key Generation. Similarly to our pre-DAA scheme, each eSIM manufacturer (EUM) randomly picks $x_0, x_1, x_2 \in_R \mathbb{Z}_p^*$ as its private keys and computes the corresponding public keys ($C_{x_0} = g^{x_0} h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \tilde{X}_2 = \tilde{h}^{x_2}$) where $\tilde{x}_0 \in_R \mathbb{Z}_p^*$.

Group Signing Key Issuance. It is analogous to the *Join* phase of our pre-DAA-A scheme. More precisely, upon its manufacture, the eSIM randomly selects $s \in_R \mathbb{Z}_p^*$. Then, it computes $C = X_1^{EID'} X_2^s$ where EID' is a 256-bit identifier derived from EID. Next, it builds the ZKPK $\pi_1 = \text{PoK}\{\alpha, \beta : C = X_1^\alpha X_2^\beta\}$ that it sends to the EUM along with C. If π_1 is valid, the EUM randomly selects $b \in_R \mathbb{Z}_p^*$ and computes (u, u') where $u = h^b$ and $u' = u^{x_0} C^b$. The eSIM is provided with the pair (u, u') as well as a ZKPK π_2 that this pair is well formed. The proof π_2 is defined as $\pi_2 = \text{PoK}\{\alpha, \beta, \gamma : u = h^\alpha \wedge u' = u^\beta C^\alpha \wedge C_{x_0} = g^\beta h^\gamma\}$. If the check of π_2 validity succeed, the pair (u, u') is stored in the eSIM as its group signing key.

Profile Generation. Following each subscription change, the Mobile Network Operator (MNO) asks a given SM-DP+ server to generate the eSIM new profile. As soon as the profile is created, the SM-DP+ must notify the SM-DS that it holds a new profile for a given eSIM. This shall be done in a privacy-preserving way (*i.e.* the SM-DS should not be able to know the EID of the relevant eSIM). To do so, the SM-DP+ computes the set $\mathsf{S} = \{\mathcal{H}(d_s)^{EID'}, \mathcal{H}(d_s+1)^{EID'}, \ldots, \mathcal{H}(d_s + Max)^{EID'}\}$ such that d_s and Max respectively correspond to the subscription date and the time period during which the new profile remains available for download (for example, $Max = 7$ for an availability of 7 days). Obviously, the subscription date and time period during which the profile is available can be set according to different time units (*e.g.* days, hours, etc.). The set S along with Add_{SM-DP+}, which denotes the address of the SM-DP+ that generated the new profile, are forwarded to the SM-DS.

Discovery request. To download its new network profile, the eSIM must first know which SM-DP+ holds it (*i.e.* the address of the SM-DP+ that has generated it). To this end, it generates an anonymous profile request that it transmits to the discovery server (SM-DS). A request initiated at a given date d_c consists of the tag $Id_c = \mathcal{H}(d_c)^{EID'}$, where $\mathsf{bsn} = d_c$, along with a signature σ on it. σ is defined as $\sigma = (w, c_1, c_2, c', V, \pi_3)$. These values are computed

Schemes	Sign	Verify
LRSW based DAA schemes		
CPS10 [16]	$7\mathbb{G}_1$	$2\mathbb{G}_1^2$, 4P
CU15-1 [17]	(7+n+u) $\mathbb{G}_1$	$2\mathbb{G}_1$, $2\mathbb{G}_1^n$, $2\mathbb{G}_1^r$, $2\mathbb{G}_1^u$, 6P
CDL16-1 [13]	$9\mathbb{G}_1$	$2\mathbb{G}_1^2$, 4P
BFGSW13 [4]	$7\mathbb{G}_1$	$2\mathbb{G}_1^2$, 4P
Our pre-DAA scheme	**$6\mathbb{G}_1$**	**$2\mathbb{G}_1^2$, 3P**
q-SDH based DAA schemes		
CF08 [18]	$3\mathbb{G}_1$, $2\mathbb{G}_T$, $2\mathbb{G}_1^2$, 1P	$1\mathbb{G}_1^2$, $2\mathbb{G}_1^3$, $1\mathbb{G}_T^5$, 3P
Ch10 [15]	$3\mathbb{G}_1$, $1\mathbb{G}_T$, $1\mathbb{G}_T^3$	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^4$, 1P
BL10 [9]	$3\mathbb{G}_1$, $1\mathbb{G}_1^2$, $1\mathbb{G}_T$, 1P	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^4$, 1P
CU15-2 [17]	$5\mathbb{G}_1$, $1\mathbb{G}_1^{2+u}$, 2P	$1\mathbb{G}_1^2$, $1\mathbb{G}_1^{4+n}$, 2P
CDL16-2 [11]	$4\mathbb{G}_1$, $1\mathbb{G}_1^{2+u}$, $1\mathbb{G}_T$, 1P	$1\mathbb{G}_1^2$, $1\mathbb{G}_2^2$, $1\mathbb{G}_T^{4+n}$, 1P
CDL16 [12]	$5\mathbb{G}_1$, $2\mathbb{G}_1^2$, $1\mathbb{G}_1^{2+u}$	$1\mathbb{G}_1^2$, $1\mathbb{G}_1^3$, $1\ G_1^{5+n}$, 2P

Table 1: DAA schemes efficiency comparison

Off-line part (card)	
Battery-On	**Battery-Off**
(240-263) 252 *ms*	(753-831) 798 *ms*

On-line part		
Signature generation (card)		*Signature verification (PC)*
Battery-On	**Battery-Off**	(5-15) 9 *ms*
(154-164) 160 *ms*	(450-472) 462 *ms*	

Total On-line part	
Battery-On	**Battery-Off**
(159-179) 169 *ms*	(455-487) 471 *ms*

Table 2: Timings ((min-max) average) in ms of the implementation of the *Sign* protocol

similarly to the *Sign* protocol of the extended version of our pre-DAA scheme, with $s_1 = EID'$, $s_2 = s$ and $T = Id_c$. Upon the receipt of a profile request, the SM-DS first verifies that $w \neq 1$ and $e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \cdot e(c_2, \tilde{X}_2) = e(c'V, \tilde{h})$. If so, it checks the proof π_3. The request is considered valid only if all checks succeed. In such case, the SM-DS queries its database to verify whether there is an entry with the same $Id_c = \mathcal{H}(d_c)^{EID'}$. If it is the case, it provides the eSIM with Add_{SM-DP+}, the address of the corresponding SM-DP+ from which the eSIM can download its new profile. Otherwise, the discovery request is discarded.

5 EFFICIENCY COMPARISON AND PERFORMANCE ASSESSMENT

In this section, we first compare the computational efficiency of our pre-DAA scheme to that of the most efficient DAA schemes. Then, we provide timings results of the implementation of the *Sign* protocol of our pre-DAA scheme on a standard, GlobalPlatform compliant, NFC SIM card.

5.1 Efficiency Comparison

In Table 1, we compare the efficiency of our scheme with that of the other DAA schemes. More precisely, we provide the total estimated cost of generating a signature as well as the computational cost of verifying it, since they are the most time critical phases.

We use the following notation: $k\mathbb{G}_i$ and $k\mathbb{G}_i^j$ respectively denote k exponentiations in the group $\mathbb{G}_i$ and k j-multi-exponentiations in the group $\mathbb{G}_i$ whereas kP refers to k pairing computations. When attributes are supported, n, r and u are used to denote the total number of attributes, the number of revealed attributes and the number of unrevealed ones respectively.

As shown in Table 1, LRSW-based DAA schemes are more efficient than $q-$SDH-based schemes. More specifically, our pre-DAA scheme has the most efficient signing operation. Furthermore, aside from [12] which was recently introduced by Camenisch *et al.*, all others $q-$SDH-based DAA schemes require the platform to perform either pairing computations or computations in $\mathbb{G}_T$ for generating a signature. Besides, all $q-$SDH and LRSW-based DAA schemes require, contrarily to our proposals, pairing computations to verify the validity of a group signing key. Thus, unlike our pre-DAA scheme, they are not suitable for SIM cards which do not support these kinds of computations. It is also worth mentioning that, as stated in [11], the ZKPKs associated to the schemes [9, 15, 17] are flawed (they do not really prove the possession of a valid group signing key). Furthermore, the security model used in [16] is flawed and the underlying DAA is not proven secure in the game-based security model introduced by Bernhard et al. [4]. Therefore, the security of the DAA scheme proposed in [16] is questionable. We would also like to emphasize that the DAA-A scheme proposed in [17] does not, contrarily to our proposal, perfectly hide the value of the group signing key attributes and should not be used in applications where these attributes may have a small entropy.

5.2 Performance Assessment

Table 2 gives timing results of the implementation of our pre-DAA scheme on a Javacard 2.2.2 SIM card, GlobalPlatform 2.2 compliant, embedded in a Samsung galaxy S5 NFC smartphone. Compared to the javacard specifications, the only particularity of our card is some additional API provided by the card manufacturer enabling operations in modular and elliptic curve arithmetic. To be able to handle asymmetric cryptography on elliptic curves, the used card is equipped with a cryptoprocessor. This makes it more powerful than most cards. It is, however, worth to emphasize that such SIM cards are already widely deployed by some phone carriers to provide NFC based services.

The implementation uses a 256-bit prime "pairing friendly" Barreto-Naehrig elliptic curve. In our implementation, the protocol is split into two parts: an *off-line* part that can be run in advance by the card and an *on-line* part that needs to be performed on-line as it depends on the verifier's challenge and basename.

Communication between the SIM card in the smartphone and the PC (Intel Xeon CPU 3.70GHz), acting as the Verifier, was done in NFC using a standard PC/SC reader (an Omnikey 5321). The *Signature generation* (by card) refers to the total time, from the applet selection to the proof reception, including the sending of the challenge Ch and the message m, which are 256-bit length, as well as the basename bsn by the verifier, but excluding the proof verification. As for the average, it is computed for 100 iterations of the sign protocol.

Battery-Off denotes a powered-off phone either by the user, or because the battery is flat. In such a situation, as stated by NFC standards, NFC-access to the SIM card is still possible, but with lower performances.

Off-line computations are assumed to be automatically launched by the smartphone (battery-On) following a proof generation and in anticipation for the next one. It is worth mentioning that all computations on the platform side are entirely performed by the card. In fact, the smartphone is only used to trigger the protocol and to power the card.

As regards to *On-line* computations, they refer to the computation of the values T, t_2 and the hash c as well as R_1 involved in the proof π_3. They can be potentially carried out even with a battery-off phone (provided that they are triggered from an external card reader).

In our implementation, we assumed that the bsn is not known in advance but rather provided to the card on-line. Consequently, the card has two scalar multiplications to compute *on-line.* For use cases where the bsn can be known in advance, these scalar multiplications may be included in the *off-line* computation part, hence resulting in much better performances[4].

6 CONCLUSION

In this paper, we introduced the most efficient and practical DAA scheme where, contrarily to previous proposals, all computations on the platform side required to either verify the validity of a group signing key or to generate a DAA are entirely performed by the TPM. Our pre-DAA scheme, which is formally proven secure, can support private-key and signature based revocations as well as group signing keys with attributes. Through the design of a privacy-preserving protocol intended for eSIM, that is built upon our pre-DAA scheme, we showed one of the potential usages of such a practical DAA scheme to address a real world issue that was raised by the GSM Association (GSMA). Finally, implementation results confirm the efficiency of our pre-DAA scheme even when implemented on a standard SIM card.

[4]One scalar multiplication on our card takes about 60 ms.

Acknowledgments.
We would like to thank the anonymous reviewers for their constructive remarks.

REFERENCES

[1] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC 2005*, pages 319–331. Springer Berlin Heidelberg, 2006.
[2] L. Batina, J.-H. Hoepman, B. Jacobs, W. Mostowski, and P. Vullers. *Developing Efficient Blinded Attribute Certificates on Smart Cards via Pairings*, pages 209–222. Springer Berlin Heidelberg, 2010.
[3] Bellare, Namprempre, Pointcheval, and Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
[4] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. In *International Journal of Information Security*, volume 12, pages 219–249, 2013.
[5] S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, pages 318–333. Springer Berlin Heidelberg, 1997.
[6] S. A. Brands. An efficient off-line electronic cash system based on the representation problem. CWI (Centre for Mathematics and Computer Science), 1993.
[7] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CCS 2004*, pages 132–145. ACM, 2004.
[8] E. Brickell and J. Li. Enhanced Privacy Id: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *WPES 2007*, pages 21–30. ACM, 2007.
[9] E. Brickell and J. Li. A pairing-based daa scheme further reducing tpm resources. In *TRUST 2010*, pages 181–195. Springer Berlin Heidelberg, 2010.
[10] E. Brickell and J. Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. In *IJIPSI*, volume 1, pages 3–33, 2011.
[11] J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *TRUST 2016*, pages 1–20. Springer International Publishing, 2016.
[12] J. Camenisch, M. Drijvers, and A. Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. Cryptology ePrint Archive, Report 2016/663, 2016.
[13] J. Camenisch, M. Drijvers, and A. Lehmann. Universally composable direct anonymous attestation. In *PKC 2016*, pages 234–264, 2016.
[14] S. Canard, B. Schoenmakers, M. Stam, and J. Traoré. List signature schemes. In *Discrete Applied Mathematics*, volume 154, pages 189–201, 2006.
[15] L. Chen. A daa scheme requiring less tpm resources. In *Inscrypt 2009*, pages 350–365. Springer Berlin Heidelberg, 2010.
[16] L. Chen, D. Page, and N. P. Smart. On the design and implementation of an efficient daa scheme. In *CARDIS 2010*, pages 223–237. Springer Berlin Heidelberg, 2010.
[17] L. Chen and R. Urian. DAA-A: Direct anonymous attestation with attributes. In *TRUST 2015*, pages 228–245. Springer International Publishing, 2015.
[18] X. Chen and D. Feng. Direct anonymous attestation for next generation TPM. In *JCP*, volume 3, pages 43–50, 2008.
[19] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, pages 418–430. Springer Berlin Heidelberg, 2000.
[20] A. de la Piedra, J.-H. Hoepman, and P. Vullers. *Towards a Full-Featured Implementation of Attribute Based Credentials on Smart Cards*, pages 270–289. Springer International Publishing, 2014.
[21] N. Desmoulins, R. Lescuyer, O. Sanders, and J. Traoré. Direct anonymous attestations with dependent basename opening. In *CANS 2014*, pages 206–221.
[22] G. AlLee, Intel. EPID for IoT Identity. https://img.en25.com/Web/McAfeeE10BuildProduction/%7Ba6dd7393-63f8-4c08-b3aa-89923182a7e5%7D_EPID_Overview_Public_2016-02-08.pdf?elqTrackId=48387d7899274c7985c6ac808d6ecbac&elqaid=7811&elqat=2, 2016.
[23] GSMA. *SGP.01 Embedded SIM Remote Provisioning Architecture, Technical Specification, V3.1*, 2014.
[24] GSMA. *SGP.02 Remote Provisioning Architecture for Embedded UICC, Technical Specification, V1.1*, 2016.
[25] GSMA. *SGP.21 RSP Architecture V2.0*, 2016.
[26] GSMA. *SGP.22 Remote SIM Provisioning Technical Specification; Version 1.1*, 2016.
[27] International Organization for Standardization. ISO/IEC 20008-2: Information technology - Security techniques - Anonymous digital signatures - Part 2: Mechanisms using a group public key, 2013.
[28] W. Lueks, G. Alpár, J.-H. Hoepman, and P. Vullers. *Fast Revocation of Attribute-Based Credentials for Both Users and Verifiers*, pages 463–478. Springer International Publishing, 2015.
[29] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *SAC'99, 1999 Proceedings*, pages 184–199. Springer Berlin Heidelberg, 2000.

[30] U. Maurer. *Cryptography and Coding: 10th IMA International Conference, 2005. Proceedings*, chapter Abstract Models of Computation in Cryptography, pages 1–12. Springer Berlin Heidelberg, 2005.
[31] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
[32] D. Pointcheval and O. Sanders. Short randomizable signatures. In *Topics in Cryptology - CT-RSA 2016*, pages 111–126. Springer International Publishing, 2016.
[33] V. Shoup. *Advances in Cryptology — EUROCRYPT '97: International Conference on the Theory and Application of Cryptographic Techniques Konstanz, Germany, May 11–15, 1997 Proceedings*, chapter Lower Bounds for Discrete Logarithms and Related Problems, pages 256–266. Springer Berlin Heidelberg, 1997.
[34] Trusted Computing Group. TPM main specification (version 1.2), 2004. http://www.trustedcomputinggroup.org/tpm-main-specification/.

A SECURITY MODEL

In this appendix, we formally define the different properties that a pre-DAA scheme must satisfy to be secure whilst preserving user's privacy [4]. For that, a pre-DAA scheme must ensure *correctness*, *traceability*, *non-frameability* as well as *anonymity*.

We follow Bernhard's *et al.* game-based model where each property is formally defined through an experiment between a challenger C and a probabilistic polynomial time (PPT) adversary $\mathcal{A}$. We distinguish two sets of users: $\mathcal{HU}$, the set of honest users and $\mathcal{CU}$, the set of corrupted users (whose secret key sk_i and group signing key gsk_i are known to the adversary). We also define three lists: $\mathbb{S}$, the list of all queries to the signing oracle; $\mathbb{C}$, the list of queries to the challenge oracle; and $\mathbb{T}$, the list of transcript views associated to the different executions of the Join protocol. To model the capabilities of the adversary $\mathcal{A}$, we give him access to some oracles defined as follows:

- $\mathcal{O}\mathsf{Add}()$ is an oracle used by $\mathcal{A}$ to create a new honest user identified by i.
- $\mathcal{O}\mathsf{AddCorrupt}()$ is an oracle used by $\mathcal{A}$ to create a new corrupted user identified by i. His secret key sk_i is known to $\mathcal{A}$.
- $\mathcal{O}\mathsf{Join}_I()$ is an oracle that executes the issuer's side of the Join protocol. This oracle will be used to simulate the execution of the Join protocol between an, honest or corrupted, user U_i and an *honest* issuer.
- $\mathcal{O}\mathsf{Join}_U(i)$ is an oracle that executes the user's side of the Join protocol. This oracle will be used by $\mathcal{A}$ acting as a malicious issuer. The adversary provides the oracle with an honest user's identifier i. If the latter accepts, $\mathcal{A}$ gets a transcript view $\mathcal{T}$ of the protocol execution, that is saved in $\mathbb{T}$.
- $\mathcal{O}\mathsf{Corrupt}(i)$ is an oracle used by $\mathcal{A}$ to corrupt the user identified by i. Thus, $\mathcal{A}$ gets both the secret key sk_i and the group signing key gsk_i of user i. Concurrently, user i is moved from $\mathcal{HU}$ to $\mathcal{CU}$.
- $\mathcal{O}\mathsf{GSign}(i, m, \mathsf{bsn})$ is an oracle used by $\mathcal{A}$ to obtain a signature on m and with respect to bsn, produced by the honest user i. Concurrently, the triple (i, m, bsn) is saved in $\mathbb{S}$.
- $\mathcal{O}\mathsf{CH}_b(i_0, i_1, \mathsf{bsn}, m)$ is an oracle used by $\mathcal{A}$ to obtain a signature σ on m generated by i_b.

For the sake of clarity, we define the following three algorithms:

- $\mathsf{Identify}_\mathsf{T}(\mathcal{T}, sk_i)$ which returns 1 if the transcript $\mathcal{T}$ resulting from the execution of the Join protocol has been produced with the secret key $sk_i = (s_1^i, s_2^i)$ (*i.e.* if C is a commitment to s_1^i and s_2^i). Otherwise, it returns 0.
- $\mathsf{Identify}_\mathsf{S}(\sigma, m, \mathsf{bsn}, sk_i)$ that returns 1 if the signature σ on the message m and with respect to bsn could have been produced using the secret key $sk_i = (s_1^i, s_2^i)$ (*i.e.* if $T = \mathcal{H}(\mathsf{bsn})^{s_1^i}$). Otherwise, it returns 0.
- $\mathsf{Link}(gmpk, \sigma, m, \sigma', m', \mathsf{bsn})$ which returns 1 if both σ and σ' are valid signatures on, respectively, m and m' with respect to the same basename $\mathsf{bsn} \neq \perp$ and were produced by the same user (*i.e.* if $T = T'$). Otherwise, it returns 0.

The four security and privacy requirements of a pre-DAA scheme are defined as follows:

Correctness. This property ensures the proper functioning of the system. A pre-DAA scheme provides correctness if the following four statements are met: (1) the issued group signing key gsk_i is valid; (2) a valid signature is accepted by the verifier $\mathcal{V}$; (3) a valid signature can be traced to the correct sk_i; and (4) two signatures produced by the same user (*i.e.* same key sk_i) and for the same bsn are linkable. Besides, it is noteworthy to mention that each transcript $\mathcal{T}$ resulting from the execution of the Join protocol must have a unique sk_i associated to it.

Traceability. Roughly speaking, traceability requires the following two statements to be met. Indeed, no adversary should be able to produce: (1) a valid signature which cannot be traced to a secret key that has been committed to during an execution of the Join protocol; or (2) two *unlinkable* signatures for the same basename and under the same secret key. Thus, as shown in Figure 3, the traceability experiment consists of two subgames. In the first subgame (*i.e.* steps 3 and 4), the issuer is assumed to be honest whereas, in the second one (*i.e.* steps 5 to 7), the issuer as well as all users are assumed to be corrupted. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{trace}}(1^k)$ of the adversary is defined as $\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{trace}}(1^k) = 1]$. A pre-DAA scheme satisfies the *traceability* property if this advantage is negligible for any PPT adversary $\mathcal{A}$.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{trace}}(1^k)$
    (1) $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{T} \leftarrow \varnothing$
    (2) $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
    (3) $(\sigma, m, \mathsf{bsn}, sk_1, \ldots, sk_l) \longleftarrow \mathcal{A}_1^{\mathcal{O}}(gmpk)$ where
        $\mathcal{O} = \{\mathcal{O}\mathsf{Add}, \mathcal{O}\mathsf{AddCorrupt}, \mathcal{O}\mathsf{Join}_I, \mathcal{O}\mathsf{Corrupt}, \mathcal{O}\mathsf{GSign}\}$
    (4) If the following 3 conditions hold, then return 1
        (a) $\mathsf{GVerify}(gmpk, \sigma, m, \mathsf{bsn}) = 1$.
        (b) $\forall \mathcal{T} \in \mathbb{T}, \exists i \in \{1, \ldots, l\}$ such that
            $\mathsf{Identify}_\mathsf{T}(\mathcal{T}, sk_i) = 1$.
        (c) $\forall i \in \{1, \ldots, l\}, \mathsf{Identify}_\mathsf{S}(\sigma, m, \mathsf{bsn}, sk_i) = 0$.
    (5) $(\sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}, sk) \longleftarrow \mathcal{A}_2(gmpk, gmsk)$
    (6) If $\mathsf{bsn} = \perp$, then return 0.
    (7) If the following 3 conditions hold, then return 1
        (a) $\forall b \in \{0, 1\}, \mathsf{GVerify}(gmpk, \sigma_b, m_b, \mathsf{bsn}) = 1$.
        (b) $\forall b \in \{0, 1\}, \mathsf{Identify}_\mathsf{S}(\sigma_b, m_b, \mathsf{bsn}, sk) = 1$.
        (c) $\mathsf{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}) = 0$.
    (8) Return 0
```

Figure 3: Traceability Security Experiment

Non-frameability. Informally, non-frameability requires the following two statements to be satisfied. Indeed, no adversary should be able to output: (1) a signature that can be traced to a given user i who has not produced a signature on the corresponding message/basename pair; or (2) two signatures that are linkable even though they should not. Thus, as shown in Figure 4, the non-frameability experiment also consists of two subgames. In the first subgame (*i.e.* steps 3 and 4), the adversary's goal is to frame an honest user whereas, in the second one (*i.e.* steps 5 to 9), he has control over all users as well as the issuer. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{non-fram}}(1^k)$ of the adversary is defined as $\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k) = 1]$. A pre-DAA scheme satisfies the *non-frameability* requirement if this advantage is negligible for any PPT adversary $\mathcal{A}$.

```
$\mathsf{Exp}_{\mathcal{A}}^{\text{non-fram}}(1^k)$
    (1)  $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{S} \leftarrow \varnothing$
    (2)  $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
    (3)  $(\sigma, i, m, \mathsf{bsn}) \longleftarrow \mathcal{A}_1^{O}(gmpk, gmsk)$ where $O = \{O\mathsf{Add}$,
         $O\mathtt{AddCorrupt}, O\mathtt{Join}_U, O\mathtt{Corrupt}, O\mathtt{GSign}\}$
    (4)  If the following 4 conditions hold, then return 1
           (a)  $\mathtt{GVerify}(gmpk, \sigma, m, \mathsf{bsn}) = 1$.
           (b)  $i \in \mathcal{HU}$.
           (c)  $(i, m, \mathsf{bsn}) \notin \mathbb{S}$.
           (d)  $\mathtt{Identify}_S(\sigma, m, \mathsf{bsn}, sk_i) = 1$.
    (5)  $(\sigma_0, m_0, \mathsf{bsn}_0, \sigma_1, m_1, \mathsf{bsn}_1, sk) \longleftarrow \mathcal{A}_2(gmpk, gmsk)$
    (6)  If $\exists b \in \{0, 1\}$ such that $\mathtt{GVerify}(gmpk, \sigma_b, m_b, \mathsf{bsn}_b) = 0$, then return 0.
    (7)  If $\forall b \in \{0, 1\}, \mathtt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}_b) = 0$, then return 0.
    (8)  If $\mathtt{Identify}_S(\sigma_0, m_0, \mathsf{bsn}_0, sk) = 1$ and
         $\mathtt{Identify}_S(\sigma_1, m_1, \mathsf{bsn}_1, sk) = 0$, then return 1.
    (9)  If $\mathsf{bsn}_0 \neq \mathsf{bsn}_1$ or $\mathsf{bsn}_0 = \perp$ or $\mathsf{bsn}_1 = \perp$, then return 1.
   (10)  Return 0
```

Figure 4: Non-Frameability Security Experiment

Anonymity. Roughly speaking, *anonymity* requires that, given two identities i_0 and i_1, no entity can determine which of the two identities produced a specific signature σ. More formally, the anonymity experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k)$ is detailed in Figure 5. The adversary queries the $O\mathsf{CH}_b$ oracle on $(i_0, i_1, \mathsf{bsn}, m)$ as input and has to guess the identity i_b of the user who generated the returned signature σ. To prevent the adversary from trivially wining the game, once he has queried the $O\mathsf{CH}_b$ oracle, he has a restricted access to the $O\mathsf{CH}_b$ and $O\mathtt{GSign}$ oracles. More precisely, he can no longer call on the $O\mathtt{GSign}$ and $O\mathsf{CH}_b$ oracles for the same (i, bsn) pair. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k)$ of the adversary is defined as $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2|$. A pre-DAA scheme is considered *anonymous* if this advantage is negligible for any PPT adversary.

```
$\mathsf{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k)$
    (1)  $pp \leftarrow \mathsf{Setup}(1^k), \mathcal{HU} \leftarrow \varnothing, \mathcal{CU} \leftarrow \varnothing, \mathbb{S} \leftarrow \varnothing, \mathbb{C} \leftarrow \varnothing$
    (2)  $(gmpk, gmsk) \leftarrow \mathsf{IKeyGen}(pp)$
    (3)  $b' \longleftarrow \mathcal{A}^{O}(gmpk, gmsk)$ where $O = \{O\mathsf{Add}, O\mathtt{AddCorrupt}$,
         $O\mathtt{Join}_U, O\mathtt{Corrupt}, O\mathtt{GSign}^*, O\mathsf{CH}_b{}^*\}$ such that $O\mathtt{GSign}^*$ and $O\mathsf{CH}_b{}^*$ de-
         note a restricted access to these oracles.
    (4)  If $i_0 \in \mathcal{CU}$ or $i_1 \in \mathcal{CU}$, then return $\perp$.
    (5)  If $\exists i, m, \sigma, \mathsf{bsn}$ such that $\mathsf{bsn} \neq \perp, (i, \mathsf{bsn}) \in \mathbb{C}$
         and $(i, m, \mathsf{bsn}) \in \mathbb{S}$, then return $\perp$.
    (6)  Return $b'$
```

Figure 5: Anonymity Security Experiment

B PROOFS OF THEOREM 1

In this appendix, we formally prove that the extended version of our pre-DAA scheme (*i.e.* the variant with one secret and one attribute) ensures the expected security properties in the random oracle model. As usual, correctness follows by inspection.

B.1 Traceability

Proof. Let $\mathcal{A}$ be an adversary who breaks the traceability requirement of our pre-DAA scheme with non-negligible probability. As previously mentioned, we distinguish the following two types of forgers:

- **Type-1 Forger:** An adversary that manages to output a signature σ that cannot be traced to a secret key previously queried to the $O\mathtt{Join}_I$ oracle.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 under the same secret sk_i and for the same basename bsn, and yet are unlinkable.

In what follows, we show that a Type-1 forger can be used as a subroutine to construct an algorithm $\mathcal{B}$ against the assumption 1, whereas Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its challenger, denoted by C, the public parameters $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, h, \tilde{h})$ as well as $X_1 = h^{x_1}$, $X_2 = h^{x_2}$, $\tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}$ and $\tilde{X}_2 = \tilde{h}^{x_2}$. As it is against the assumption 1, $\mathcal{B}$ has access to the oracle O_1 which, on input $m_1, m_2 \in \mathbb{Z}_p$, outputs the pair $P = (t, t^{x_0+m_1x_1+m_2x_2})$ where $t \in_R \mathbb{G}_1$[5]. This oracle O_1 will be used during $O\mathtt{Join}_I()$ requests. $\mathcal{B}$ chooses the generator g and the value of C_{x_0} at random. Thereby, it can provide $\mathcal{A}$ with the public parameters pp of the scheme and answers his (*i.e.* $\mathcal{A}$'s) requests as follows:

- $O\mathsf{Add}(i)$ requests: $\mathcal{B}$ creates a new user i and picks two random values $s_1, s_2 \in_R \mathbb{Z}_p^*$ as his secret key sk_i. $\mathcal{B}$ also generates user's i public/private endorsement key pair denoted by (esk_i, epk_i).
- $O\mathtt{AddCorrupt}(i)$ requests: $\mathcal{B}$ does nothing.
- $O\mathtt{Join}_I()$ requests: Upon the receipt of C, the proof π_1 and the signature S. $\mathcal{B}$ first checks the validity of both S and π_1. If so, it uses the soundness property of π_1 to recover s_1 and s_2. Then, it queries the O_1 oracle on s_1 and s_2 to obtain the pair $(u, u' = u^{x_0+s_1x_1+s_2x_2})$. Besides, in the Random Oracle Model (ROM), π_2 can be perfectly simulated using standard techniques. Thereby, $\mathcal{B}$ can answer $\mathcal{A}$'s request by providing him with the pair (u, u') as well as the proof π_2.
- $O\mathtt{Corrupt}(i)$ requests: $\mathcal{B}$ provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.
- $O\mathtt{GSign}(i, m, \mathsf{bsn})$ requests: As it holds both sk_i and gsk_i, $\mathcal{B}$ can generate a signature on m and for bsn, that it returns to $\mathcal{B}$.

Remark: Suppose that the set S of keys sk_i revealed by $\mathcal{A}$ during its attack (step 3) is different from **S'**, which consists of the keys recovered by $\mathcal{B}$ during $O\mathtt{Join}_I$ requests. This would therefore imply that there exists two pairs of keys (s_1, s_2) and (s_1', s_2') such that $X_1^{s_1} X_2^{s_2} = X_1^{s_1'} X_2^{s_2'}$. $\mathcal{A}$ could therefore be used to break the binding property of Pedersen's commitments which relies on the discrete logarithm assumption. Thereby, $\mathcal{A}$ could be used to break

[5]Note that we use an extended version of assumption 1 that supports several messages [32]

the DL assumption and consequently to break assumption 1. Let us thus assume that $\mathsf{S} = \mathsf{S}'$.

Eventually, $\mathcal{A}$ outputs with non-negligible probability a valid triple $(\sigma, m, \mathsf{bsn})$ such that the signature σ was produced using an sk_i that is not associated to any of the calls to the $\mathcal{O}\mathtt{Join}_I$ oracle. Using the soundness of π_3, $\mathcal{B}$ extracts s_1 and s_2 as well as $w' = c'g^{-z_3}$. Thus, $\mathcal{B}$ obtains a valid pair $(w, w' = w^{x_0+s_1x_1+s_2x_2})$ on the pair (s_1, s_2) that has never been queried to the $\mathcal{O}_1$ oracle. Consequently, using $\mathcal{A}$, $\mathcal{B}$ can break the assumption 1.

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs $(\sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}, sk)$ such that σ_0 and σ_1 are valid signatures on respectively m_0 and m_1, and for the same basename bsn. Besides, the two signatures were produced using sk (as, by definition, both $\mathsf{Identify_S}(\sigma_0, m_0, \mathsf{bsn}, sk)$ and $\mathsf{Identify_S}(\sigma_1, m_1, \mathsf{bsn}, sk)$ output 1). Owing to the completeness of the proof π_3, the tags T_0 and T_1 associated to respectively σ_0 and σ_1 are necessarily defined as $T_0 = \mathcal{H}(\mathsf{bsn})^{s_1}$ and $T_1 = \mathcal{H}(\mathsf{bsn})^{s_1}$. Therefore, we have $T_0 = T_1$. By definition, $\mathcal{A}$ wins if the output of $\mathsf{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn})$ is 0 (*i.e.* $T_0 \neq T_1$). Thus, such forgery can never occur.

Therefore, if $\mathcal{A}$ can break the traceability property of our pre-DAA scheme, then $\mathcal{B}$ can break the assumption 1 with the same probability. Thus, under the assumption 1, our pre-DAA scheme provides the traceability requirement in the ROM.

□

B.2 Non-frameability

Proof. Let $\mathcal{A}$ be an adversary against the non-frameability requirement of our pre-DAA scheme. As previously mentioned in appendix A, we distinguish the following two types of forgers:

- **Type-1 Forger:** An adversary that manages to output a signature σ on m and for bsn that is traced to a specific user that has never produced such a signature.
- **Type-2 Forger:** An adversary that outputs two signatures σ_0 and σ_1 that are linkable even though they should not (*i.e.* they were either produced with different sk_i or with respect to two distinct basenames).

In what follows, we show that a Type-1 forger can be used to construct a reduction $\mathcal{B}$ against the OMDL assumption whilst a Type-2 forgery cannot happen. Initially, $\mathcal{B}$ chooses a random bit $c_{mode} \in \{1, 2\}$ that indicates its guess for the type of forgery that $\mathcal{A}$ will output.

If $c_{mode} = 1$: $\mathcal{B}$ receives on input from its challenger C a random instance $(h^{u_1}, h^{u_2}, \ldots, h^{u_n})$ of the OMDL problem where h is a random generator of $\mathbb{G}_1$. As it is against the one-more discrete logarithm OMDL assumption, $\mathcal{B}$ has access to a DL oracle. The adversary $\mathcal{A}$ picks three random values $x_0, x_1, x_2 \in_R \mathbb{Z}_p^*$ as the issuer's private key and publishes the associated public key $gmpk = (C_{x_0} = g^{x_0}h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \tilde{X}_2 = \tilde{h}^{x_2})$ where $\tilde{x}_0 \in_R \mathbb{Z}_p^*$. $\mathcal{B}$ uses the soundness property of π to recover x_0, $\tilde{x}_0$, x_1 and x_2 and answers $\mathcal{A}$'s requests as follows:

- $\mathcal{O}\mathsf{Add}(i)$ requests: $\mathcal{B}$ creates a new user U_i and randomly selects $s_2^i \in_R \mathbb{Z}_p^*$. Using its input of the OMDL problem, it sets s_1^i as the discrete logarithm of $C_i = h^{s_1^i} = h^{u_i}$ in the base h. The user's secret key is defined as $sk_i = (s_1^i, s_2^i)$ where s_1^i is unknown to both $\mathcal{B}$ and $\mathcal{A}$. $\mathcal{B}$ also generates U_i's public/private endorsement key pair denoted by (esk_i, epk_i).
- $\mathcal{O}\mathtt{AddCorrupt}$ requests: $\mathcal{B}$ does nothing.
- $\mathcal{O}\mathtt{Join}_U(i)$ requests: $\mathcal{B}$ computes $C = C_i^{x_1}X_2^{s_2^i} = X_1^{u_i}X_2^{s_2^i} = X_1^{s_1^i}X_2^{s_2^i}$.
  Since $\mathcal{B}$ holds esk_i, then it can compute the signature S. As for π_1, $\mathcal{B}$ can perfectly simulate it in the random oracle model. If the protocol does not abort, $\mathcal{B}$ obtains a valid group signing key $gsk_i = (u, u')$ associated to $sk_i = (s_1^i, s_2^i)$, where s_1^i is unknown to both $\mathcal{A}$ and $\mathcal{B}$, as well as the proof π_2. Using the soundness property of π_2, $\mathcal{B}$ retrieves the value of b (required to simulate the $\mathcal{O}\mathtt{GSign}$ oracle).
- $\mathcal{O}\mathtt{Corrupt}(i)$ requests: $\mathcal{B}$ calls on the DL oracle with h^{u_i} as input. Thereby, it recovers $u_i = s_1^i$. Thus, it can provide $\mathcal{A}$ with the secret key $sk_i = (s_1^i, s_2^i)$ along with the group signing key gsk_i of user i.
- $\mathcal{O}\mathtt{GSign}(i, m, \mathsf{bsn})$ requests: As it holds s_2^i and gsk_i, $\mathcal{B}$ can compute most of the required values except c_1, T and the proof π_3. Using b, which was extracted from π_2 during the run of Join_U, $\mathcal{B}$ sets $c_1 = C_i^{br_1}h^{z_1} = w^{s_1^i}h^{z_1}$ where $r_1, z_1 \in_R \mathbb{Z}_p^*$ such that $w = u^{r_1}$. To compute T, $\mathcal{B}$ proceeds as follows: it randomly selects $l \in_R \mathbb{Z}_p^*$ and sets $H = \mathcal{H}(\mathsf{bsn}) = h^l$. Thus, $T = \mathcal{H}(\mathsf{bsn})^{u_i} = C_i^l$. During subsequent calls to $\mathcal{O}\mathtt{GSign}$ on input the same i and bsn, $\mathcal{B}$ returns the same value T. As for π_3, it can be easily simulated in the ROM. Hence, $\mathcal{B}$ can perfectly simulate the $\mathtt{GSign}$ algorithm.

Eventually, after d calls to the $\mathcal{O}\mathtt{Corrupt}$ oracle, $\mathcal{A}$ outputs with non negligible probability a valid signature σ on m and for bsn, such that $\mathsf{Identify_S}(\sigma, m, \mathsf{bsn}, sk_i)$ outputs 1 whereas the user holding sk_i has never produced a signature on that m and for bsn. By definition of the experiment, we know that the corresponding user is *honest*. Thus, the u_i associated to the user's *unknown* secret s_1^i is still in the input of the OMDL problem. Using the soundness property of π_3, $\mathcal{B}$ retrieves the associated secrets s_1^i and s_2^i where $s_1^i = u_i$. Thus, $\mathcal{B}$ recovers u_i, the discrete logarithm of the challenge h^{u_i}. By outputting u_i along with the d secrets $\{u_j\}_{j=1}^{j=d}$ that it has obtained by querying the DL oracle, $\mathcal{B}$ breaks the OMDL assumption.

If $c_{mode} = 2$: $\mathcal{A}$ eventually outputs two valid signatures σ_0 and σ_1, on respectively m_0 and m_1, that are linkable even though they should not. That is, they were either generated using two different keys (sk_0 and sk_1) or/and are for different basenames (bsn_0 and bsn_1). Let $sk_b = (s_1^b, s_2^b)$ denote the key used to generate σ_b and $sk = (\tilde{s}_1, \tilde{s}_2)$ the key output by $\mathcal{A}$. If $\mathcal{A}$'s attack against the non-frameability property is successful, then this implies that condition 6 of the non-frameability experiment is true. In particular, owing to the completeness of π_3, we have

$$T_0 = \mathcal{H}(\mathsf{bsn}_0)^{s_1^0} \text{ and } T_1 = \mathcal{H}(\mathsf{bsn}_1)^{s_1^1} \quad (1)$$

The condition 7 should also be true. Thus, $\exists b \in \{0,1\}$ such that $\mathtt{Link}(gmpk, \sigma_0, m_0, \sigma_1, m_1, \mathsf{bsn}_b) = 1$. Without loss of generality, let us assume that $b = 0$. From the validity of this assertion, we can deduce that

$$T_0 = \mathcal{H}(\mathsf{bsn}_0)^{s_1^0} = T_1 = \mathcal{H}(\mathsf{bsn}_0)^{s_1^1} \quad (2)$$

This implies that

$$s_1^0 = s_1^1 \; (mod\; p) \quad (3)$$

For $\mathcal{A}$'s forgery to be successful, condition 8 or 9 should be true. Suppose that assertion 8 is true. This implies that

$$T_0 = \mathcal{H}(\mathsf{bsn}_0)^{\tilde{s}_1} \quad (4)$$

and

$$T_1 \neq \mathcal{H}(\mathsf{bsn}_1)^{\tilde{s}_1} \quad (5)$$

This is impossible. Indeed, (2) and (4) imply that $s_1^0 = \tilde{s}_1$. Thus, from (3), we have $s_1^0 = \tilde{s}_1 = s_1^1$. From (1), we know that $T_1 = \mathcal{H}(\mathsf{bsn}_1)^{s_1^1} = \mathcal{H}(\mathsf{bsn})^{\tilde{s}_1}$, which contradicts (5).

Let us now assume that condition 9 is true, *i.e.* that $\mathsf{bsn}_0 \neq \mathsf{bsn}_1$. (1) and (2) imply that $T_1 = \mathcal{H}(\mathsf{bsn}_1)^{s_1^1} = \mathcal{H}(\mathsf{bsn}_0)^{s_1^1}$, hence $\mathcal{H}(\mathsf{bsn}_0) = \mathcal{H}(\mathsf{bsn}_1)$ where $\mathsf{bsn}_0 \neq \mathsf{bsn}_1$. This would imply that $\mathcal{A}$ broke the *second pre-image resistance* property of the hash function $\mathcal{H}$ which is infeasible in the ROM.

Therefore, a Type-2 forgery can never occur.

Consequently, under the OMDL assumption, our pre-DAA scheme ensures the non-frameability requirement.

□

B.3 Anonymity

Proof. Let $\mathcal{A}$ be an adversary against the anonymity requirement of our pre-DAA scheme. Using Shoup's game hopping technique, where proofs are organized as sequences of games, we prove that our pre-DAA scheme ensures the anonymity requirement under the XDH assumption. Hereinafter, we provide a high level description of the initial game (Game 0) along with a brief description of the transition between Game 0 and Game 1.

Game 0: It corresponds to the real attack game with respect to an efficient adversary $\mathcal{A}$.

The Challenger C randomly chooses the system public parameters $pp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \mathcal{H}, e)$ as well as the issuer's private key $gmsk = (x_0, x_1, x_2)$ and computes the associated public key $gmpk = (C_{x_0} = g^{x_0}h^{\tilde{x}_0}, X_1 = h^{x_1}, X_2 = h^{x_2}, \tilde{X}_0 = \tilde{h}^{x_0}, \tilde{X}_1 = \tilde{h}^{x_1}, \tilde{X}_2 = \tilde{h}^{x_2})$ where $h \in_R \mathbb{G}_1$ and $\tilde{x}_0 \in_R \mathbb{Z}_q$. Then, it provides $\mathcal{A}$ with $gmsk = (x_0, x_1, x_2)$ and answers his requests as follows:

- $\mathcal{O}\mathsf{Add}(i)$ requests: C creates a new user i and picks two random value $s_1^i, s_2^i \in_R \mathbb{Z}_p^*$ as his secret key sk_i. C also generates user's i public/private endorsement key pair denoted by (esk_i, epk_i).
- $\mathcal{O}\mathtt{AddCorrupt}(i)$ requests: C does nothing.
- $\mathcal{O}\mathtt{Join}_U(i)$ requests: C computes the commitment C, then builds π_1 to obtain the group signing key gsk_i.
- $\mathcal{O}\mathtt{Corrupt}(i)$ requests: C provides $\mathcal{A}$ with the secret key sk_i and the group signing key gsk_i of user i.
- $\mathcal{O}\mathtt{GSign}(i, m, \mathsf{bsn})$ requests: C uses sk_i and gsk_i to generate a signature σ on m and for bsn, that it returns to $\mathcal{A}$.

Eventually, $\mathcal{A}$ chooses $(i_0, i_1, \mathsf{bsn}, m)$ that he provides as input to $\mathcal{O}\mathsf{CH}_b$. The oracle outputs a signature σ produced by user i_b. $\mathcal{A}$'s goal is to guess the value of b. Even after receiving his challenge, $\mathcal{A}$ can still query the $\mathcal{O}\mathsf{Add}$, $\mathcal{O}\mathtt{AddCorrupt}$, $\mathcal{O}\mathtt{Join}_U$, $\mathcal{O}\mathtt{Corrupt}$ and $\mathcal{O}\mathtt{GSign}$ oracles but with some restrictions. More precisely, he cannot call the $\mathcal{O}\mathtt{Corrupt}$ oracle on i_0 or i_1. Besides, he is not allowed to query the $\mathcal{O}\mathtt{GSign}$ oracle with either (i_0, bsn) or (i_1, bsn) as input. Otherwise, he would trivially win the game.

So, $\mathcal{A}$ eventually outputs his guess b'. Let S_0 be the event that $b = b'$ in Game 0 (*i.e.* the event that $\mathcal{A}$ wins Game 0) and S_1 denote the event that $b = b'$ in Game 1. We have $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = |\Pr[S_0] - 1/2|$. We construct the Game 1 as follows:

Game 1: It is the same game as Game 0 except that we replace $T = \mathcal{H}(\mathsf{bsn})^{s_1^b}$ by a random value and simulate the proof π_3. Such a proof can be easily simulated in the ROM using classical techniques. Under the XDH assumption, $\mathcal{A}$ cannot detect this change. In fact, one can easily construct a XDH distinguisher $\mathcal{D}$ with an advantage of solving the XDH problem, denoted by $\mathsf{Adv}_{\mathsf{XDH}}$, satisfying $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathsf{XDH}}(1^k)$. In Game 1, the challenger provides no information (in a strong information theoretic sense) about the bit b to the adversary $\mathcal{A}$. This is due to the *perfectly hiding* property of Pedersen's commitment. Let us consider a valid signature (w, c_1, c_2, c', V). For any valid group signing key $gsk_i = (\tilde{u}, \tilde{u}')$ and corresponding secrets $\tilde{s}_1^l$, $\tilde{s}_2^l$, there exists $\tilde{l}$, $\tilde{z}_1, \tilde{z}_2, \tilde{z}_3$ such that $w = \tilde{u}^{\tilde{l}}, w' = (\tilde{u}')^{\tilde{l}}, c_1 = w^{\tilde{s}_1^l}h^{\tilde{z}_1}, c_2 = w^{\tilde{s}_2^l}h^{\tilde{z}_2}$ and $c' = w'g^{\tilde{z}_3}$. Let us show that this implies that $V = g^{-\tilde{z}_3}X_1^{\tilde{z}_1}X_2^{\tilde{z}_2}$. Since (w, c_1, c_2, c', V) is a valid signature, then we have:

$$e(w, \tilde{X}_0) \cdot e(c_1, \tilde{X}_1) \cdot e(c_2, \tilde{X}_2) = e(c'V, \tilde{h}) \quad (6)$$

This implies that

$$e(w^{x_0}w^{\tilde{s}_1^l x_1}h^{\tilde{z}_1 x_1}w^{\tilde{s}_2^l x_2}h^{\tilde{z}_2 x_2}, \tilde{h}) = e(w'g^{\tilde{z}_3}V, \tilde{h})$$

Therefore,

$$w^{x_0 + \tilde{s}_1^l x_1 + \tilde{s}_2^l x_2}h^{\tilde{z}_1 x_1}h^{\tilde{z}_2 x_2} = w'g^{\tilde{z}_3}V \quad (7)$$

Since (w, w') is a valid group signing key for the secret keys $\tilde{s}_1^l$ and $\tilde{s}_2^l$, this implies that

$$w' = w^{x_0 + \tilde{s}_1^l x_1 + \tilde{x}_2^l x_2} \quad (8)$$

(7) and (8) implies that $V = g^{-\tilde{z}_3}h^{\tilde{z}_1 x_1}h^{\tilde{z}_2 x_2} = g^{-\tilde{z}_3}X_1^{\tilde{z}_1}X_2^{\tilde{z}_2}$.

In Game 1, the only value involving the user's secret key and computed in a non-perfectly hiding way (*i.e.* $T = \mathcal{H}(\mathsf{bsn})^{s_1^b}$) was replaced by a random value. Therefore, $\Pr[S_1] = 1/2$. Thus, we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k) = |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{anon}-b}(1^k) = b] - 1/2| = \\ |\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathsf{XDH}}(1^k)$$

Consequently, if the XDH assumption holds, $\mathsf{Adv}_{\mathsf{XDH}}(1^k)$ will be negligible. This implies that the adversary's advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{anon}-b}(1^k)$ will also be negligible. Thus, we conclude that our pre-DAA scheme satisfies the anonymity requirement under the XDH assumption.

□