# Direct Anonymous Attestation from Lattices

Rachid El Bansarkhani and Ali El Kaafarani
TU Darmstadt, Germany
University of Oxford, UK

***Abstract*—Direct Anonymous Attestation (DAA) is a complex cryptographic protocol that has been widely deployed in practice, with more than 500 million machines in the market that are already equipped with its hardware, the so-called Trusted Module Platform (TPM). While formalizing the right security model for such a complex protocol has triggered a dense line of research, all the proposed DAA schemes so far are based on number-theoretic problems that are known to be vulnerable to quantum computer attacks. In this paper, we propose the first lattice-based DAA scheme that is secure w.r.t. the most up-to-date security model proposed by Camenisch et al. More precisely, our lattice-based DAA scheme is secure in the Universally Composable (UC) security model. Furthermore, we give (amongst others) the first lattice-based DAA scheme providing user controlled linkability that is realized by means of a new lattice-based MAC/TAG construction which could be of independent interest.**

**Keywords: Lattice-Based Cryptography, Direct Anonymous Attestation, Universally Composable security model.**

## I. Introduction

Direct Anonymous Attestation (DAA) was first introduced in [8] by Brickell, Camenisch and Chen[1]. It is a protocol that allows a certain machine to prove that it has the right system configuration, while communicating with other machines but without revealing any further details about itself. In fact, it is a small chip embedded in a host machine, which can anonymously attest to the system's configurations of the host to remote verifiers. For instance, it can prove that the host machine is built of certified hardware and is indeed running the correct software. The small chip is called the Trusted Platform Module (TPM) and was standardized by the Trusted Computing Group (TCG) in 2004 as TPM 1.2 [1], and in 2014 as TPM2.0 [2] respectively. The attestation can be considered a special type of anonymous digital signatures that reveals no information about the identity of the signer, which in this case is the TPM itself. The user of the host system can at discretion choose whether or not she wants her transactions directed at the same verifier to be linkable. This property is often called the user-controlled linkability. The "attestations", that are directed at the same verifier, may involve the same basename of this verifier and therefore make the transactions attached to those attestations linkable, i.e. the verifier would then know that he is communicating with the same anonymous host. However, attestations with different basenames or empty ones are guaranteed to be non-linkable. Although DAA is a complex cryptographic protocol, it has been widely deployed in practice, where more than 500 millions[2] of computers in the market are already equipped with TPM chips. It can therefore be considered as one of the most deployed complex cryptographic protocols. This encouraged a strong research interest in improving the security and efficiency of the protocol and this shows in the huge body of work related to DAA [11], [9], [10], [15], [18], [5], [13], [12].

The journey of formalizing the security model of such a complex scheme was not smooth; the first scheme proposed in [8] came along with a simulation-based security definition. The ideal functionality was restricted to a single procedure, i.e. no separation between the signature generation and the verification. Additionally, linkability was not explicitly defined. The linkability issue was addressed in [16], whereas an attempt at fixing the problem of the functionality in the security model was made later on in [17], in which they managed to separate between the signature generation and the verification, but unfortunately the way they got separated caused problems in both of them. The line of simulation-based work stopped here, to resume later on by the work of Camenisch el. work in [13], which will be explained thoroughly in this paper as this is the most up-to-date security model and that is why we adopted it in our work. However, the gap between the simulation-based work was filled with property-based security models. It started with Brickell, Chen, and Li in [10], in which they defined two security games, namely, the anonymity and user-controlled traceability. Unfortunately, the games as they were defined didn't capture all the possible threat scenarios, even after adding the non-frameability game in [15].

All the shortcomings in the existing game-based DAA schemes were discussed and analyzed by Bernard et al. in [5]. Additionally, they propose a new game-based security model that works in two phases; in the first phase, they deal with the host and TPM as a single entity (platform), i.e. they are always in the same corruption state. They call this scheme pre-DAA. This is not a problem when it comes to proving the anonymity and the non-frameability, as both the host and TPM are assumed to be honest, whereas the issuer is assumed to be corrupt. However, things get more complicated when it comes to unforgeability. The main challenge in the DAA design is to put more workload on the side of the more powerful part of the platform, which is the host in this case. However, this should not affect the security of the DAA protocol in case the host is corrupt. We want the unforgeability property to hold even when the TPM is honest and the host is corrupt. In the second phase of the DAA construction in [5], they (informally) show how to achieve a full-DAA from a pre-DAA, but due to the lack of a formal security model for the full DAA, Camenisch el al. in [13] point out that the proof of the pre-DAA unforgeability cannot be lifted to the full-DAA

[1]This paper was published in ACM-CCS 2004, and won the *Test-of-Time* award in 2014.

[2]http://www.trustedcomputinggroup.org/authentication/

case under the same assumption. This problem was addressed in [13] but in the Universal Composable (UC) model this time. The advantages of the UC framework over the stand-alone one (i.e. game-based one) are clear. First, no need to define oracles as the adversary gets full control over corrupt parties. But more importantly, the universal composition theorem of the UC model [14] states that if a given scheme is secure in the UC model, then it can be composed in an arbitrary way without affecting its security.

Lattice-based cryptography gained much attention as a result of seminal works presented in [3], [27], [29], [25], [7], which introduced the major computational problems (Ring-)SIS and (Ring-)LWE serving as the fundamental assumptions for almost all lattice-based cryptographic primitives. These problems are characterized by efficient instantiations and worst-case to average-case relationships preventing from major faults in the protocol design. For instance, they were exploited to build sophisticated schemes such as anonymous digital signatures in [20], [22], [23], [28], [24], which are of interest for a great variety of applications.

### A. Contributions and Technical Overview

We propose the first direct anonymous attestation scheme from lattice assumptions and prove its security in the UC model w.r.t. the most up-to-date security model defined in [13]. Therefore, not only were we able to answer in the affirmative the question of whether or not it is possible to build a DAA protocol based on post-quantum assumptions, but also managed to prove it secure following the most practically-interesting security model for DAAs. In the following paragraphs we briefly highlight the main technical contributions of our work.

*1) Lattice-Based MAC Scheme.:* We introduce a new lattice-based MAC scheme from (Ring-)LWE that is unforgeable under chosen message attacks in the random oracle model. For a message $m$ and randomness $\mu$, the MAC is generated as

$$\mathsf{MAC}((\mathsf{msg}, \mu), \mathbf{s}) = H(\mathsf{msg}, \mu) \cdot \mathbf{s} + \mathbf{e}$$

for a shared secret $\mathbf{s}$ and error vector $\mathbf{e}$. Due to its deterministic behavior, i.e. it always outputs almost the same element for the same $(\mathsf{msg}, \mu)$, it can serve as a tag in other lattice-based constructions.

*2) CMA-Secure Digital Signature Scheme.:* Given our MAC scheme, we show that we can build a digital signature scheme using a zero-knowledge proof system. Briefly speaking, the encryption of the shared secret will serve as the public key, i.e.

$$\mathsf{pk} = (\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}) .$$

Now, we can link the secret of any output of $\mathsf{MAC}(\cdot, \mathbf{s})$ with $\mathsf{pk}$ via a zero-knowledge proof system which together yield publicly verifiable signatures. To this end, we consider the (Stern-like) statistical non-interactive zero-knowledge argument of knowledge (sZKAoK) due to [24], which allows to prove the possession of small secrets in (Ring-)LWE and (Ring-)SIS instances.

*3) DAA Construction.:* Finally, we transform this signature scheme into a DAA scheme providing anonymity to all signers, if $\mathsf{pk}$ is a (Ring-)SIS instance with many secrets $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$ satisfying $\mathbf{u} = \mathbf{A}_{\mathsf{id}} \cdot \mathbf{z} \bmod q$, and user controlled linkability via the MAC scheme with secret $\mathbf{z}_1$. In fact, when instantiating our construction with Boyen's signature scheme [6], the signature size is only logarithmic in the number of signers $N$, i.e. the signature and public key occupy only $\log N \cdot \tilde{O}(n)$ bits (ring variant) or $\log N \cdot \tilde{O}(n^2)$ bits (matrix variant) of memory, which is comparable to the most efficient lattice-based group signature schemes. Our construction can be seen as a group signature scheme without tracing capabilities, but with an additional tag that is added to the SPK. The tag plays a crucial role in our DAA scheme as the verifier can request from a certain signer to sign with respect to a basename $\mathsf{bsn}$ such that all his/her so produced signatures directed at this verifier will then be anonymously linkable (i.e. they stem from the same anonymous signer), that is because $\mathsf{tag} = \mathsf{MAC}(\mathsf{bsn}, \mathbf{s})$ is almost deterministic, i.e. $\|\mathsf{tag} - H(\mathsf{bsn}) \cdot \mathbf{s}\| \leq \beta$.

This feature equips the verifier with a powerful *partial* signature-revocation mechanism, i.e. the verifier can revoke signatures that prove linkable to a *bad* signature (e.g. in case of misuse/abuse). However, this is only a partial mechanism as it doesn't cover the non-linkable signatures. More specifically, if the signer does not want to link his signatures to each other, he can simply set the basename to $\perp$ so that his signatures become completely anonymous, and therefore the signature-revocation mechanism doesn't work as such. On the other hand, the tags come along with a *full* secret key-revocation mechanism, i.e. once the secret key of a certain TPM is extracted and made public, it is possible via the tag to find and revoke all signatures that are generated by use of this secret key.

More importantly, one of the main challenges in the DAA design is to allow the TPM and host located on a platform to *jointly* generate a signature that is anonymous, but in such a way that each one *only* knows its secret share. This separation should also be reflected in the security model (i.e. allow them to be in different corruption states), and as we mentioned earlier, this separation problem was completely solved only recently in [13]. In our scheme, we tackle this challenge this way; we let the TPM generate a small secret vector $\mathbf{z}_1$ and a proof of knowledge that $\tilde{\mathbf{u}} = \mathbf{A}_{\mathsf{id}} \cdot \mathbf{z}_1 \bmod q$ and $\|\mathbf{z}_1\| \leq \beta$, where $\mathbf{A}_{\mathsf{id}} = [\hat{\mathbf{A}} \mid \hat{\mathbf{A}}_0 + \sum_{i=1}^{\ell} \mathsf{id}_i \cdot \hat{\mathbf{A}}_i]$ is a matrix associated to the identity $\mathsf{id} \in \{0,1\}^{\ell}$ of the TPM. The host will send this proof to the issuer who registers both $\tilde{\mathbf{u}}$ and the corresponding TPM, and responds with a small secret share $\mathbf{z}_2$ such that $\mathbf{u} = \mathbf{A}_{\mathsf{id}} \cdot (\mathbf{z}_1 + \mathbf{z}_2) \bmod q$. Using a zero-knowledge proof system, we build a signature scheme with *distributed secrets*. That is, the TPM and host generate a signature without revealing their secret shares by generating the proofs $\pi_1, \pi_2$ separately, where $\pi_1$ proves $\tilde{\mathbf{u}} = \mathbf{A}_{\mathsf{id}} \cdot \mathbf{z}_1 \bmod q$ for a small $\mathbf{z}_1$ and $\pi_2$ proves $\mathbf{u} - \tilde{\mathbf{u}} = \mathbf{A}_{\mathsf{id}} \cdot \mathbf{z}_2 \bmod q$ for a small $\mathbf{z}_2$. Taking $\pi_1$ and $\pi_2$ together, proves that $\mathbf{z}_1 + \mathbf{z}_2$ is a valid credential satisfying $\mathbf{u} = \mathbf{A}_{\mathsf{id}} \cdot (\mathbf{z}_1 + \mathbf{z}_2) \bmod q$ and $\|\mathbf{z}_1 + \mathbf{z}_2\| \leq 2\beta$. This shows that proofs for distributed secrets can be combined in order to sign. However, at this

stage the signatures are still not anonymous since $\mathbf{u} - \tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}$ are unique for each platform. Thus, we re-randomize the signature, i.e. the TPM generates a small random vector $\mathbf{t}$ for each signature such that $\mathbf{A}_{\mathsf{id}} \cdot \mathbf{t} \bmod q$ is uniform random. This vector is then shared among the TPM and host (via a secure channel). They can now generate proofs $\pi_1, \pi_2$ with a new and uniform random $\mathbf{u}'$, where $\pi_1$ proves $\mathbf{u}' = \mathbf{A}_{\mathsf{id}} \cdot (\mathbf{z}_1 + \mathbf{t}) \bmod q$ for a small $\mathbf{z}_1 + \mathbf{t}$ and analogously $\pi_2$ proves $\mathbf{u} - \mathbf{u}' = \mathbf{A}_{\mathsf{id}} \cdot (\mathbf{z}_2 - \mathbf{t}) \bmod q$ for a small $\mathbf{z}_2 - \mathbf{t}$.

*4) Security.:* We prove security of our construction in the UC model using the most recent framework of Camenisch et al. [13] for DAA, where we require online extractability which is assured, for instance, by Unruh's transformation [30] in order to obtain a non-interactive proof system. However, in a stand-alone fashion we can use rewinding, (e.g. by use of Fiat-Shamir) leading to a more efficient scheme.

*5) Organization.:* We present the preliminaries in Section 2 and the building blocks in Section 3. We then present the security model in Section 4. Finally, we present our DAA construction together with the security proofs in Section 5.

## II. Preliminaries

We will use the polynomial rings $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ such that $f(X)$ is a monic and irreducible polynomial over $\mathcal{R}$ and $q$ denotes the modulus. Throughout this paper we will mainly consider the case $q = 2^k,\ k > \mathbb{N}$. For the Ring-LWE problem we mainly focus on cyclotomic polynomials $f(X) = X^n + 1$ for $n$ being a power of 2. We denote ring elements by boldface lower case letters e.g. $\mathbf{p}$, whereas for vectors of ring elements we use $\hat{\mathbf{p}}$. By $[k]$ we denote the set of integers $\{1, \ldots, k\}$ and $[k]_0$ if it also contains 0. Throughout this work we can either use $\|\cdot\|_2$ or $\|\cdot\|_\infty$ .

### A. Lattices

A $k$-dimensional lattice $\Lambda$ is a discrete additive subgroup of $\mathbb{R}^m$ containing all integer linear combinations of $k$ linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ with $k \leq m$ and $m \geq 0$. More formally, we have $\Lambda = \{\ \mathbf{B} \cdot \mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^k\ \}$.

The discrete Gaussian distribution over a non-empty set $\Lambda$ denoted by $D_{\Lambda,s}$ assigns to $\mathbf{x} \in \Lambda$ the probability $e^{-\pi \frac{\|\mathbf{x}\|^2}{s^2}} / \sum_{\mathbf{v}\in\Lambda} e^{-\pi \frac{\|\mathbf{v}\|^2}{s^2}}$.

Below we give a description of the Ring-LWE distribution and the related problems.

### B. Computational Problems

*Definition 2.1 (Ring-LWE Distribution):* Let $n, q$ be integers and $\chi_e$ be the error distribution over $\mathcal{R}$. By $L^{\mathsf{RLWE}}_{n,\alpha q}$ we denote the Ring-LWE distribution over $\mathcal{R}_q \times \mathcal{R}_q$, which draws $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q$ uniformly at random, samples $\mathbf{e} \xleftarrow{\$} \chi_e$ and returns $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}_q \times \mathcal{R}_q$ for $\mathbf{s} \in \mathcal{R}_q$ and $\mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$.

*Definition 2.2 (Ring-LWE Problem):* Let $\mathbf{u}$ be uniformly sampled from $\mathcal{R}_q$.

- The decision problem of Ring-LWE asks to distinguish between $(\mathbf{a}, \mathbf{b}) \leftarrow L^{\mathsf{RLWE}}_{n,\alpha q}$ and $(\mathbf{a}, \mathbf{u})$ for a uniformly sampled secret $\mathbf{s} \xleftarrow{\$} \mathcal{R}_q$.
- The search problem of Ring-LWE asks to return the secret vector $\mathbf{s} \in \mathcal{R}_q$ given a Ring-LWE sample $(\mathbf{a}, \mathbf{b}) \leftarrow L^{\mathsf{RLWE}}_{n,\alpha q}$ for a uniformly sampled secret $\mathbf{s} \xleftarrow{\$} \mathcal{R}_q$ ($\mathbf{s}$ can also be sampled from the error distribution).

## III. Building Blocks

This section is devoted to the main ingredients required to construct our DAA scheme.

### A. Non-Interactive Zero Knowledge Argument

Throughout this work, we need to generate proofs for the possession of valid credentials and the knowledge of short secrets. Consider the following language;

$$\begin{aligned} L \quad = \quad & \{\mathsf{public} := \{\mathbf{b}, \hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u}, \mathsf{msg}, \mu\} \\ & \mathsf{witness} := \{\mathsf{id}, \hat{\mathbf{z}} = (\mathbf{z}_0, \ldots, \mathbf{z}_{2m})\} : \\ & \{\hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{z}} = \mathbf{u} \wedge \|\hat{\mathbf{z}}\| \leq \beta\} \\ & \wedge\{\mathsf{nym} = H(\mathsf{msg}, \mu) \cdot \mathbf{z}_0 + \mathbf{e} \wedge \|\mathbf{e}\| \leq \beta\}\}, \end{aligned}$$

where $\hat{\mathbf{A}}_{\mathsf{id}} = [\hat{\mathbf{A}} \mid \hat{\mathbf{A}}_0 + \sum_{i=1}^{\ell} \mathsf{id}_i \cdot \hat{\mathbf{A}}_i\ ]$. Such a language can be proven in zero-knowledge using the statistical zero-knowledge argument of knowledge (sZAoK) due to [24] instantiated with the commitment scheme of Kawachi et al. [21]. The sZAoK satisfies the standard properties completeness, zero-knowledge and special soundness. It can finally be made non-interactive via the Fiat-Shamir heuristic or other non-interactive proof systems, which are online extractable [19].

### B. Boyen's Digital Signature Scheme

In the following section we give a short description of Boyen's signature scheme [6] instantiated in the ring setting as it represents one of the main ingredients of our DAA construction. In particular, we consider $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for $\mathcal{R} = \mathbb{Z}[X]/\langle X^n + 1\rangle$ and $n = 2^k$. Thus, let $m = O(\log q)$ for a prime $q \equiv 1 \bmod 2n$. Denote by $2^\ell$ the number of possible messages, which will later on be replaced by the secret identities.

1) $\mathsf{Gen}(1^n)$: Generate a set of polynomials $\hat{\mathbf{A}} \in \mathcal{R}_q^m$ endowed with a trapdoor $\mathbf{T}$ by e.g. [26], [4]. Furthermore, sample uniform random sets of polynomials $\hat{\mathbf{A}}_i \in \mathcal{R}_q^m$ for $i \in [\ell]_0$. Finally, select a uniform random syndrome $\mathbf{u} \in \mathcal{R}_q$. The secret and public keys are given by $\mathsf{sk} := \mathbf{T}$ and $\mathsf{pk} := (\hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u})$.
2) $\mathsf{Sign}(\mathsf{sk}, \mathsf{id} \in \{0,1\}^\ell)$: First, generate the vector of polynomials $\hat{\mathbf{A}}_{\mathsf{id}} = [\hat{\mathbf{A}} \mid \hat{\mathbf{A}}_0 + \sum_{i=1}^{\ell} \mathsf{id}_i \cdot \hat{\mathbf{A}}_i\ ]$ associated to $\mathsf{id}$. Subsequently, sample polynomials $\hat{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_{2m}) \hookleftarrow D_{\Lambda^\perp_{\mathbf{u}}(\hat{\mathbf{A}}_{\mathsf{id}}),s}$ satisfying $\hat{\mathbf{A}}_{\mathsf{id}} \cdot \mathbf{z} \equiv \mathbf{u} \bmod q$ by use of the trapdoor $\mathbf{T}$. Finally, output the signature $\hat{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_{2m})$.
3) $\mathsf{Verify}(\mathsf{pk}, \mathsf{id} \in \{0,1\}^\ell, \mathbf{z})$: If $\hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{z}} \equiv \mathbf{u} \bmod q$ and $\|\hat{\mathbf{z}}\| \leq \beta$ are satisfied, output 1, else 0.

The scheme is based on the hardness of (Ring-)SIS and is proven to be secure in the standard model. We refer to [6] for

a security proof, which has been improved later on in [26] by use of a new trapdoor and its ring analogue.

*Remark.* We note that a signature $\hat{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_{2m})$ satisfying $\mathbf{u} = \hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{z}}$ can be turned into a solution $\bar{\mathbf{z}} = (\hat{\mathbf{a}}, \hat{\mathbf{b}}, \mathsf{id}_1 \cdot \hat{\mathbf{b}}, \ldots, \mathsf{id}_\ell \cdot \hat{\mathbf{b}})$ of an Ring-ISIS instance $\bar{\mathbf{A}} \cdot \bar{\mathbf{z}} = \mathbf{u} \bmod q$ with $\|\bar{\mathbf{z}}\| \leq \beta$, where $\bar{\mathbf{A}} = [\hat{\mathbf{A}} \mid \hat{\mathbf{A}}_0 \mid \hat{\mathbf{A}}_1 \mid \ldots \mid \hat{\mathbf{A}}_\ell]$ and $\hat{\mathbf{a}} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$ and $\hat{\mathbf{b}} = (\mathbf{z}_{m+1}, \ldots, \mathbf{z}_{2m})$.

### C. A New MAC-based Signature Scheme from Lattices

Much progress has been made to construct efficient lattice-based signature schemes that have recently even become practical. In fact, one differentiates between random oracle variants and constructions that are provably secure in the standard model. For instance, the lines of works based on either the Fiat-Shamir transform or the GPV signature scheme are considered to be the most common representatives. However, many of those constructions are limited with respect to their features. Therefore, we present in this section a new lattice-based signature scheme, that is interesting with respect to its properties rather than its performance as it requires a non-interactive statistical zero-knowledge argument of knowledge (NIZKAoK) system to provide unforgeability of the scheme. Once there are efficient instantiations of NIZKAoK protocols, it immediately carries over to the signature scheme. From a high level view, we can consider our digital signature scheme to be composed by a lattice-based symmetric encryption scheme in combination with a new lattice-based MAC algorithm on the message and a statistical zero-knowledge argument of knowledge NIZKAoK for the possession of the secret.

We now propose a new (stateful) MAC scheme retaining security based on the (Ring-)LWE problem. We start with a generic description of the algorithms and the related security model. Subsequently, we will present our construction followed by a security proof.

*Definition 3.1 (**CMA secure MAC**):* A MAC is a pair of algorithms $\mathcal{M} = (\mathsf{MAC}, \mathsf{Verify}_{\mathsf{MAC}})$ with $\mathsf{MAC} : A_1 \times A_2 \to B$ and $\mathsf{Verify}_{\mathsf{MAC}} : A_1 \times A_2 \times B \to \{0,1\}$ is a CMA-secure MAC, if:

- **Validity** For every $\mathsf{msg} \in A_1$, $\mathbf{s} \in A_2$ $\mathsf{Verify}_{\mathsf{MAC}}(\mathsf{msg}, \mathbf{s}, \mathsf{MAC}(\mathsf{msg}, \mathbf{s})) = 1$.
- **Security** For every polynomially bounded adversary the advantage of the adversary

$$\mathsf{Adv}^{CMA}_{\mathcal{M},\mathcal{A}}(n) = P[\mathsf{Exp}^{CMA}_{\mathcal{M},\mathcal{A}}(n) = 1] \leq \epsilon$$

in the experiment $\mathsf{Exp}^{CMA}_{\mathcal{M},\mathcal{A}}(k, N)$ is negligible.

**Experiment:** $\mathsf{Exp}^{CMA}_{\mathcal{M},\mathcal{A}}(n)$

- $\mathbf{s} \leftarrow \mathsf{KeyGen}_{\mathsf{MAC}}(1^n)$
- $(\mathsf{msg}^*, \tau^*, \mathsf{St}) \leftarrow \mathcal{A}^{\mathsf{OMAC}(\mathbf{s},\cdot),\mathsf{OVerify}_{\mathsf{MAC}}(\mathbf{s},\cdot,\cdot)}$
- If $\mathsf{msg}^*$ was queried to OMAC, return 0.
- Return 1 if $\mathsf{Verify}_{\mathsf{MAC}}(\mathbf{s}, \mathsf{msg}^*, \tau^*) = 1$, else 0.

In the following sections, we define $\beta = \alpha q \cdot \omega(1)$ for a distribution $\chi = D_{\mathbb{Z},\alpha q}$ such that $\underset{x \sim D_{\mathbb{Z},\alpha q}}{P}[|x| > \beta]$ is negligible.

**Construction of a stateful MAC scheme.** Let $H(\cdot) \leftarrow \mathcal{R}_q$ denote a cryptographic hash function modeled as random oracle. Our MAC scheme is defined by 3 algorithms $\mathcal{M} = (\mathsf{KeyGen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify}_{\mathsf{MAC}})$.

1) $\mathsf{KeyGen}_{\mathsf{MAC}}(1^n)$ : On input security parameter $n$, it generates a secret polynomial $\mathbf{s} \hookleftarrow \chi^n = D_{\mathbb{Z}^n,\alpha q}$. The key space is $\mathcal{K} = [-\beta, \beta]^n \cap \mathbb{Z}^n$. Output $\mathsf{sk} := \mathbf{s}$.
2) $\mathsf{MAC}(\mathsf{msg}, \mathsf{sk})$ : If the local storage contains a record $(\mathsf{msg}, \tau_{\mathsf{msg}})$ output $\tau_{\mathsf{msg}}$, otherwise generate $\mathbf{b} = H(\mathsf{msg}) \in \mathcal{R}_q$ and compute $\tau_{\mathsf{msg}} = \mathbf{b} \cdot \mathbf{s} + \mathbf{e}$ for an error vector $\mathbf{e} \hookleftarrow \chi^n$. Finally, store the record $(\mathsf{msg}, \tau_{\mathsf{msg}})$ in the local storage and output the tag $\tau_{\mathsf{msg}}$.
3) $\mathsf{Verify}_{\mathsf{MAC}}(\mathsf{msg}, \mathsf{sk}, \tau)$ : On input the secret key, a message and a tag, it invokes $\tau^* = \mathsf{MAC}(\mathsf{msg}, \mathsf{sk})$ and checks if $\|\tau^* - \tau\| \leq \sqrt{2}\beta$, where $\beta$ denotes the bound related to $\chi$ such that $\underset{x \sim \chi}{P}[|x| > \beta]$ is negligible.

**Construction of a probabilistic MAC scheme.** Let $H(\cdot) \leftarrow \mathcal{R}_q$ be as above, then the algorithms are defined as follow.

1) $\mathsf{KeyGen}_{\mathsf{MAC}^*}(1^n)$ : Invoke $\mathbf{s} \leftarrow \mathsf{KeyGen}_{\mathsf{MAC}}(1^n)$. Output $\mathsf{sk} := \mathbf{s}$.
2) $\mathsf{MAC}^*(\mathsf{msg}, \mathsf{sk})$ : Sample random $\ell$-bit string $\mu \hookleftarrow \{0,1\}^\ell$ and generate $\mathbf{b} = H(\mathsf{msg}, \mu) \in \mathcal{R}_q$. Subsequently sample an error vector $\mathbf{e} \hookleftarrow \chi^n$ and compute $\tau = \mathbf{b} \cdot \mathbf{s} + \mathbf{e}$. Output the tag $\tau$ and random string $\mu$.
3) $\mathsf{Verify}_{\mathsf{MAC}^*}((\mathsf{msg}, \mu), \mathsf{sk}, \tau)$ : Compute $\tau^* = \mathsf{MAC}((\mathsf{msg}, \mu), \mathsf{sk})$ and check if $\|\tau^* - \tau\| \leq \sqrt{2}\beta$, then output 1, else 0.

*Lemma 3.2:* Let $q$ be a prime number and $t \in \mathbb{Z}_q^\times$ a prime with $\chi^n = \mathcal{D}_{\mathbb{Z}^n,\alpha q}$ such that $t \geq 2\beta$ for $\beta = \alpha q \omega(1)$ and $2t\beta < q$ (identifying $\mathcal{R}$ as $\mathbb{Z}^n$ via the coefficient embedding). For $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \hookleftarrow \chi^n$ and any $\mathbf{a} \in \mathcal{R}_q$ let $\mathbf{b}_1 = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}_1$ and $\mathbf{b}_2 = (\mathbf{a} - t) \cdot \mathbf{s} + \mathbf{e}_2$, then

$$\mathbf{s} = t^{-1} \cdot [\mathbf{b}_1 - \mathbf{b}_2 - (\mathbf{b}_1 - \mathbf{b}_2 \bmod t)] .$$

*Proof.* A straightforward calculation shows that the equality $[\mathbf{b}_1 - \mathbf{b}_2 - (\mathbf{b}_1 - \mathbf{b}_2 \bmod t)] = t \cdot \mathbf{s}$ is satisfied, since $t \geq 2\beta$ □

*Theorem 3.3:* Let $q$ be a prime number and $t \in \mathbb{Z}_q^\times$ a prime such that $t \geq 2\beta$ and $t^2 + t < q$, where $\chi^n = \mathcal{D}_{\mathbb{Z}^n,\alpha q}$. Assuming the hardness of the Ring-LWE problem, then the MAC scheme $\mathcal{M} = (\mathsf{KeyGen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Verify}_{\mathsf{MAC}})$ is CMA-secure in the random oracle model.

*Proof.* Suppose there exists an efficient adversary against the CMA-security of the MAC scheme, then we can build an efficient algorithm $\mathcal{B}$ solving the search version of Ring-LWE. Let $q_H$ denote the maximum number of hash function, $\mathsf{MAC}(\cdot)$ and $\mathsf{Verify}_{\mathsf{MAC}}(\cdot)$ queries of $\mathcal{A}$. Algorithm $\mathcal{B}$ is given $q_H$ many Ring-LWE samples $\{(\mathbf{a}_i, \mathbf{b}_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i)\}_{i=0}^{q_H}$ as the problem instance, where $\mathbf{s}, \mathbf{e}_i \hookleftarrow \chi^n$ and $\mathbf{a}_i \hookleftarrow \mathcal{R}_q$, and he is challenged to output the unknown secret vector $\mathbf{s}$. Following $\mathsf{Exp}^{CMA}_{\mathcal{M},\mathcal{A}}(n)$ the adversary is given access to the

oracles $\mathsf{OMAC}(\mathbf{s},\cdot)$ and $\mathsf{OVerify}_{\mathsf{MAC}}(\mathbf{s},\cdot,\cdot)$. Furthermore, he is given access to the random oracle $H$. We can safely assume that the adversary calls the random oracle before outputting a valid forgery. Thus, $\mathcal{B}$ selects an index $i^* \in \{1,\ldots,q_H\}$ for the critical query. The challenger simulates the environment as follows.

- **Setup** The challenger maintains a counter, which is set to $c := 0$ at the beginning of the game. He also maintains a list $H_1$, which is successively filled with triples $(\mathsf{msg},\mathbf{h},\mathbf{b})$ during the game.
- **Queries to RO.** On queries msg to the RO the challenger checks, if $H_1$ contains an entry $(\mathsf{msg},\mathbf{h},\mathbf{b})$ and outputs $\mathbf{h}$, if it exists, else he adds the entry $(\mathsf{msg},\mathbf{a}_c,\mathbf{b}_c)$, outputs $\mathbf{a}_c$ and sets $c := c+1$. If the number of entries in the $H_1$ list is $i^*-1$, $\mathcal{B}$ adds the entry $(\mathsf{msg},\mathbf{a}_{i^*}-t,\mathbf{b}_{i^*})$ to the list and outputs $\mathbf{a}_{i^*}-t$.
- **Queries to the Signing Oracle.** On queries to $\mathsf{OMAC}(\mathbf{s},\cdot)$ on messages msg, $\mathcal{B}$ will call the hash oracle on msg. If the $H_1$ list contains an entry $(\mathsf{msg},\mathbf{h},\mathbf{b})$, then he outputs $\mathbf{b}$. If it has not been set, he takes a new instance and adds $(\mathsf{msg},\mathbf{a}_c,\mathbf{b}_c)$ to the list, outputs $\mathbf{b}_c$ and sets $c := c+1$. However, if the adversary queries $\mathsf{OMAC}(\mathbf{s},\cdot)$ on the critical message at his $i^*$-th call, $\mathcal{B}$ aborts.
- **Queries to the Verification Oracle.** For all non-critical queries, message msg and tag $\mathbf{b}$ the algorithm $\mathcal{B}$ will invoke $\mathbf{b}_j \leftarrow \mathsf{OMAC}(\mathbf{s},\mathsf{msg})$ for some $j$ and output 1 if: $\|\mathbf{b}-\mathbf{b}_j\| \leq \sqrt{2}\beta$, else output 0. In case it is the critical query, it verifies that $\|\mathbf{b}-\mathbf{b}_{i^*}\| = \|t\cdot\mathbf{s}+\mathbf{e}-\mathbf{e}^*\| \leq t\|\mathbf{s}\|+\|\mathbf{e}-\mathbf{e}^*\| \leq t^2+t$, then $\mathcal{B}$ computes $\mathbf{s}$ following Lemma 3.2 applied on $\mathbf{b}$ and $\mathbf{b}_{i^*}$ and verifies that $\mathbf{b}_{i^*} = \mathbf{a}_{i^*}\cdot\mathbf{s}+\mathbf{e}_{i^*}$ for $\|\mathbf{e}_{i^*}\| \leq \beta$. If satisfied, $\mathcal{B}$ outputs $\mathbf{s}$ to the challenger. Otherwise $\mathcal{B}$ aborts.

The challenger perfectly simulates the environment. Eventually, $\mathcal{A}$ outputs a valid forgery $(\mathsf{msg}^*,\mathbf{b}^*)$ on a message that has not been queried to OMAC before. Then, $\mathcal{A}$ has either invoked the RO or not. In case the random oracle has not been invoked on $\mathsf{msg}^*$, $\mathcal{B}$ aborts. Otherwise, he must have queried the RO. But then he must have produced an instance $\mathbf{b}^* = (\mathbf{a}_{i^*}-t)\cdot\mathbf{s}+\mathbf{e}^*$ with $\mathbf{s} = t^{-1}\cdot[\mathbf{b}^*-\mathbf{b}_{i^*}-(\mathbf{b}^*-\mathbf{b}_{i^*} \bmod t)]$ and $\|\mathbf{e}^*-\mathbf{e}_{i^*}\| \leq t$ following Lemma 3.2, where $\mathbf{b}_{i^*} = \mathbf{a}_{i^*}\cdot\mathbf{s}+\mathbf{e}_{i^*}$. In this case, $\mathcal{B}$ breaks the search version of Ring-LWE, which contradicts the hardness assumption of Ring-LWE. □

The proof of Theorem 3.3 particularly also shows, that even given many samples from the Ring-LWE distribution using the same secret, the adversary cannot learn anything about the MAC outputs. Our digital signature scheme takes advantage of this fact and can thus be built from the following primitives $(\mathsf{Enc},\mathsf{MAC},\mathsf{NIZKAoK})$. Roughly speaking, the public key is an encryption of the secret (Ring-LWE instance), the MAC algorithm outputs signatures by use of this secret, and NIZKAoK is applied to deduce a proof of knowledge of a secret binding the signature to the public key.

**From MAC to a Digital Signature Scheme**. We start with the construction of our digital signature scheme. It consists of the following algorithms $\mathcal{DS} = (\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$. To remove the property of stateful signatures, the signer always samples a new seed $\mu$ in the signing step in order to randomize the output. This relieves the signer from storing already signed messages such that he can sign the same message several times. Let $H(\cdot) : \{0,1\}^* \to \mathcal{R}_q$ be a cryptographic hash function modeled as random oracle.

1) $\mathsf{KeyGen}(1^n)$ : On input security parameter $n$, generate a public parameter uniformly at random $\mathbf{a} \hookleftarrow \mathcal{R}_q$, a secret polynomial $\mathbf{s} \hookleftarrow \chi^n$ and an error polynomial $\mathbf{r} \hookleftarrow \chi^n$. Output the public key $\mathsf{pk} := \mathbf{a}\cdot\mathbf{s}+\mathbf{r}$ and secret key $\mathsf{sk} := \mathbf{s}$.

2) $\mathsf{Sign}(\mathsf{sk},\mathsf{msg})$ : Sample a seed $\mu \hookleftarrow \{0,1\}^l$ uniformly at random and generate $\tau = \mathsf{MAC}(\mathsf{msg},\mathsf{sk})$. Finally, generate

$$\begin{aligned} \pi &= \mathsf{SPK}\{\mathsf{public} := \{\mathbf{a},\mathsf{pk},\mathsf{msg},\mu,\tau\} \\ &\quad\ \mathsf{witness} := \{\mathbf{s}\} : L_{LWE}\} \\ \text{where } L_{LWE} &= \{\mathsf{pk} = \mathbf{a}\cdot\mathbf{s}+\mathbf{r} \wedge (\|\mathbf{s}\|,\|\mathbf{r}\| \leq \beta)\} \wedge \\ &\quad\ \{\tau = H(\mathsf{msg},\mu)\cdot\mathbf{s}+\mathbf{e} \wedge \|\mathbf{e}\| \leq \beta\} \end{aligned}$$

to prove that pk and $\tau$ are ciphertexts of the same secret key. Output the signature

$$\Sigma = (\mathsf{msg},\mu,\tau,\pi)$$

on message msg using randomness $\mu$.

3) $\mathsf{Verify}(\mathsf{pk},\Sigma)$ : Parse $\Sigma$ as $(\mathsf{msg},\mu,\tau,\pi)$ and verify the NIZKAoK $\pi$ on input $\mathsf{pk},\mathsf{msg},\mu$, and $\tau$. If $\pi$ is a valid proof output 1, else 0.

**Security**. Roughly speaking, based on the Ring-LWE problem an attacker cannot find the secret out of the public key and many (different) MACs, which indeed just represent Ring-LWE samples. Thus, we now state the unforgeability theorem.

*Theorem 3.4:* If SPK is a statistical non-interactive zero-knowledge argument of knowledge (sNIZKoA) for the secret in the digital signature scheme $\mathcal{DS}$. Then, $\mathcal{DS}$ is unforgeable under chosen message attacks (CMA-secure) in the RO model assuming the hardness of solving the Ring-LWE problem.

*Proof (sketch).* The signature scheme $\mathcal{DS}$ can be forged by forging the MAC scheme, which we proved in Theorem 3.3 to be secure under the (Ring-)LWE assumption, or by generating a valid SPK proof linking the public key to a MAC on a message $(\mathsf{msg},\mu)$ that has not been signed before. However, in the latter case the adversary must have broken the soundness property of the SPK, i.e. the simulator can extract the secret key by rewinding. □

**Linkable Indistinguishable Tags**. From the above constructions, we observe that a fixed signer always generates an almost identical $\bar{\tau} = H(\mathsf{msg},\mu)\cdot\mathbf{s}+\bar{\mathbf{e}}$, if he signs the same message using the same randomness, because $\|\tau-\bar{\tau}\| = \|\mathbf{e}-\bar{\mathbf{e}}\| \leq \sqrt{2}\beta$. This deterministic behavior allows the deduction of new features for certain applications related to anonymous attestation protocols. In fact, $\tau$ acts as a tag for a certain message under the same randomness, but does not reveal anything about the secret key based on the Ring-LWE assumption. For instance, if $\tau$ and $\bar{\tau}$ have been produced

**Experiment:** $\mathsf{Exp}^{f-\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(n)$
- $(\mathsf{sk}_0, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}_{\mathsf{LIT}}(1^n)$
- $\mathbf{c} \leftarrow f(\mathsf{sk}_0)$
- $b \xleftarrow{\$} \{0,1\}$
- $(\mathsf{msg}^*, \mathsf{St}) \leftarrow \mathcal{A}^{\mathsf{TAG}_{\mathsf{LIT}}(\cdot,\mathsf{sk}_0),H(\cdot)}(\mathbf{c})$
- $\tau^* \leftarrow \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}^*, \mathsf{sk}_b)$
- $b^* \leftarrow \mathcal{A}^{\mathsf{TAG}_{\mathsf{LIT}}(\cdot,\mathsf{sk}_0)}(\tau^*, \mathsf{St})$
- If $\mathsf{msg}^*$ was asked of $\mathsf{TAG}_{\mathsf{LIT}}$ return 0.
- Return 1 if $b^* = b$, else 0.

**Experiment:** $\mathsf{Exp}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(n)$
- $(\mathsf{msg}_0, \tau_0, \mathsf{sk}_0, \mathsf{msg}_1, \tau_1, \mathsf{sk}_1) \leftarrow \mathcal{A}(1^n)$
- Return 1 only if:
- $\mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}_0, \mathsf{sk}_0) = \tau_0$ for a suitable $\mathbf{e}_0$, i.e. $\|\mathbf{e}_0\| = \|H(\mathsf{msg}_0) \cdot \mathsf{sk}_0 - \tau_0\| \leq \beta$
- $\mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}_1, \mathsf{sk}_1) = \tau_1$ for a suitable $\mathbf{e}_1$, i.e. $\|\mathbf{e}_1\| = \|H(\mathsf{msg}_1) \cdot \mathsf{sk}_1 - \tau_1\| \leq \beta$
- $\mathsf{LINK}_{\mathsf{LIT}}(\tau_0, \tau_1) = 1$
- Either $\mathsf{sk}_0 \neq \mathsf{sk}_1$ or $\mathsf{msg}_0 \neq \mathsf{msg}_1$.

by different signers, the difference would be computationally indistinguishable from a uniform random polynomial.

Linkable indistinguishable tags (LIT) for classical DAA protocols have been introduced in [5] and are akin to regular message authentication codes (MAC). Below we give the related LIT algorithms in accordance to [5], however with a slight modification to handle randomized algorithms applied in our setting. More specifically, we are outputting only a disguised tag involving an error term sampled from a narrow noise distribution. Thus, the verification algorithm will check two tags with respect to approximate equality only. The generic syntax of LITs is given in [5]. Below we instantiate a LIT with our MAC scheme.

**LIT Instantiation.** A LIT is defined by a tuple of algorithms $\mathcal{S}_{\mathsf{LIT}} = (\mathsf{KeyGen}_{\mathsf{LIT}}, \mathsf{LINK}_{\mathsf{LIT}}, \mathsf{TAG}_{\mathsf{LIT}})$:

- $\mathsf{KeyGen}_{\mathsf{LIT}}(1^n)$ : Output $\mathsf{sk} \leftarrow \mathsf{KeyGen}_{\mathsf{MAC}}(1^n)$.
- $\mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}, \mathsf{sk})$ : Compute $\tau = \mathsf{MAC}(\mathsf{msg}, \mathsf{sk})$ and output the tag $\tau$.
- $\mathsf{LINK}_{\mathsf{LIT}}(\tau_0, \tau_1)$ : If $\|\tau_0 - \tau_1\| \leq \sqrt{2}\beta$ output 1, else 0.

We now define the different security notions to be satisfied. As opposed to MAC schemes, LIT doesn't necessarily require unforgeability. We slightly modified the security model for linkability as it suffices to check in the verification step of our construction that two tags are only approximately the same.

Given a one-way function applied to the secret key $\mathsf{pk} := f(\mathsf{sk})$ acting as the public key, the security of the LIT is defined relative to pk. The adversary can break a LIT scheme either via its indistinguishability property or its linkability property.

*Theorem 3.5 ($f-\mathsf{IND}$):* Let $q$ be an integer and $f(\mathbf{s}) = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \bmod q$ for uniform random polynomial $\mathbf{a} \in \mathcal{R}_q$ and $\mathbf{e} \hookleftarrow \chi^n$. In the RO model, if there exists an efficient adversary $\mathcal{A}$ against the indistinguishability property of the Ring-LWE-LIT scheme $\mathcal{S}_{\mathsf{LIT}}$ as defined above, then there exists an efficient algorithm $\mathcal{B}$ solving decision Ring-LWE$_{n,\alpha q}$ such that

$$\mathsf{Adv}^{f-\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(n) \leq q_H \mathsf{Adv}^{R-DLWE}_{\mathcal{P},\mathcal{B}}(n),$$

where $q_H$ denotes the maximum number of hash function, tag and verify queries of $\mathcal{A}$.

*Proof.* Let $\{(\mathbf{a}_i, \mathbf{b}_i = \mathbf{a}_i \cdot \mathsf{sk}_0 + \mathbf{e}_i)\}_{i=0}^{q_H-2}, (\mathbf{u}_1, \mathbf{u}_2)$ denote the input problem instance to algorithm $\mathcal{B}$, where $(\mathbf{u}_1, \mathbf{u}_2)$ is either a Ring-LWE instance or a uniform random sample. The aim of $\mathcal{B}$ is to break decision Ring-LWE$_{n,\alpha q}$, i.e. to determine if $\mathbf{u}_2 = \mathbf{u}_1 \cdot \mathsf{sk}_0 + \mathbf{e}$ or $(\mathbf{u}_1, \mathbf{u}_2)$ are uniform random samples in $\mathcal{R}_q \times \mathcal{R}_q$. At the beginning $\mathcal{B}$ calls $\mathcal{A}$ on $\mathbf{c} = f(\mathsf{sk}_0) = \mathbf{a}_0 \cdot \mathsf{sk}_0 + \mathbf{e}_0$ . The adversary is given access to $\mathsf{TAG}_{\mathsf{LIT}}(\cdot, \mathsf{sk}_0)$ and the random oracle $H(\cdot) \in \mathcal{R}_q$ following the security game in $\mathsf{Exp}^{f-\mathsf{IND}}_{\mathsf{LIT},\mathcal{A}}(n)$. Suppose that the adversary makes at most $q_H$ calls to the hash function and tag oracles, then we can safely assume that $\mathsf{msg}^*$ has been queried at the $i^*$-th call with $i^* \in \{1, \ldots, q_H\}$ before the first stage of $\mathcal{A}$ ends. Furthermore, we can assume that he invokes the RO before every call to the tag or verify oracle.

- **Queries to RO.** Algorithm $\mathcal{B}$ maintains a counter $c := 1$ and a list $H_1$ consisting of triples $(\mathsf{msg}, \mathbf{h}, \mathbf{r})$. If $H_1$ has been queried on a message msg, the challenger checks for an existing entry $(\mathsf{msg}, \mathbf{h}, *) \in H_1$ and outputs $\mathbf{h}$. If the list contains $i^* - 1$ elements, then $\mathcal{B}$ adds the challenge $(\mathsf{msg}, \mathbf{u}_1, \mathbf{u}_2)$ to the list and responds with $\mathbf{u}_1$ to $\mathcal{A}$, otherwise he adds $(\mathsf{msg}, \mathbf{a}_c, \mathbf{b}_c)$ to the list, returns $\mathbf{a}_c$ to $\mathcal{A}$ and sets $c := c + 1$.
- **TAG Queries.** In case the adversary requests a tag on a message msg, we can assume to have triples $(\mathsf{msg}, \mathbf{h}, \mathbf{r})$ in the $H_1$-list. Then, $\mathcal{B}$ returns $\mathbf{r}$ to $\mathcal{A}$.

If the adversary has not queried the hash oracle at its $i^*$-th call on the message $\mathsf{msg}^*$ returned by the adversary after the first stage, then $\mathcal{B}$ aborts. Otherwise he returns $\tau^* = \mathbf{u}_2$ as the supposed tag on $\mathsf{msg}^*$. After the second stage of the algorithm the adversary will make its guess on whether $\tau^*$ is valid, i.e. $\mathsf{sk}_0$ has been used to generate the tag $\tau^*$ on message $\mathsf{msg}^*$. The tag is only valid if $|\tau^* - (H(\mathsf{msg}^*) \cdot \mathsf{sk}_0 + \mathbf{e})| \leq \sqrt{2}\beta$, where $H(\mathsf{msg}^*) = \mathbf{u}_1$. Algorithm $\mathcal{B}$ responds with the output of $\mathcal{A}$, which is a solution to decision Ring-LWE$_{n,\alpha q}$. □

*Lemma 3.6:* Let $\mathcal{R}_q$ be defined as above. Let $\mathbf{a}_1, \ldots, \mathbf{a}_{m-1} \hookleftarrow \mathcal{R}_q$ be uniform random polynomials. Suppose there exists a PPT algorithm $\mathcal{S}$ that solves $\|\mathbf{a}_1 \cdot \mathbf{s}_1 + \ldots + \mathbf{a}_{m-1} \cdot \mathbf{s}_{m-1}\| \leq \gamma$ for non-zero $\mathbf{s}_i$, s.t. $\|\mathbf{s}_i\| \leq \gamma$ and $1 \leq i \leq (m-1)$, then there exists a PPT algorithm $\mathcal{B}$ that solves Ring-SIS$_{n,m,\gamma}$.

*Proof.* Suppose there exists such an algorithm. Define by $[\mathbf{a}_1, \ldots, \mathbf{a}_m]$ the problem instance. $\mathcal{B}$ is challenged to solve the R-SIS problem $\mathbf{a}_1 \cdot \mathbf{s}_1 + \ldots + \mathbf{a}_m \cdot \mathbf{s}_m \equiv \mathbf{0} \bmod q$ for $\|\mathbf{s}_i\| \leq \gamma$ and $1 \leq i \leq m$. Assume that one of the ring elements is invertible, which exists with non-negligible probability. Without l.o.g. assume that $\mathbf{a}_m \in \mathcal{R}_q^\times$. Then define by $[\mathbf{a}'_1 := \mathbf{a}_m^{-1}\mathbf{a}_1, \ldots, \mathbf{a}'_{m-1} := \mathbf{a}_m^{-1}\mathbf{a}_{m-1}]$ the input to $\mathcal{S}$, which outputs $\mathbf{s}_i$ for $1 \leq i \leq (m-1)$ in polynomial time such that $\|\mathbf{s}_i\| \leq \gamma$ and $\|\mathbf{a}'_1 \cdot \mathbf{s}_1 + \ldots + \mathbf{a}'_{m-1} \cdot \mathbf{s}_{m-1}\| \leq \gamma$. Algorithm $\mathcal{B}$ outputs then $[\mathbf{s}_1, \ldots, \mathbf{s}_m := -(\mathbf{a}'_1 \cdot \mathbf{s}_1 + \ldots + \mathbf{a}'_{m-1} \cdot \mathbf{s}_{m-1})]$ as a solution to the problem instance. This concludes the proof. □

*Theorem 3.7 (*LINK*):* In the RO model, if there exists a PPT adversary $\mathcal{A}$ breaking the linkability property of the RLWE-LIT, then there exists an efficient algorithm $\mathcal{B}$ solving Ring-SIS$_{n,m,\gamma}$ for $\gamma = 2\sqrt{2}\beta$ such that

$$\mathsf{Adv}^{\mathsf{LINK}}_{\mathsf{LIT},\mathcal{A}}(n) \leq q_H^2 \mathsf{Adv}^{R-SIS}_{\mathcal{P},\mathcal{B}}(n),$$

where $q_H$ denotes the maximum number of hash function calls of $\mathcal{A}$.

*Proof.* We prove that given an PPT adversary $\mathcal{A}$, that generates valid $(\mathsf{msg}_0, \tau_0, \mathsf{sk}_0, \mathsf{msg}_1, \tau_1, \mathsf{sk}_1)$ such that $\tau_0 - \tau_1 \in [-\sqrt{2}\beta, \sqrt{2}\beta]$, where $\tau_i = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}_i, \mathsf{sk}_i)$ for $i = 0, 1$, we can build an algorithm $\mathcal{B}$ to solve Ring-SIS$_{n,2,\gamma}$ or Ring-SIS$_{n,3,\gamma}$. In order to win the game, the adversary can choose $\mathsf{msg}_i$ and $\mathsf{sk}_i$ such that either $\mathsf{sk}_0 \neq \mathsf{sk}_1$ or $\mathsf{msg}_0 \neq \mathsf{msg}_1$ holds. We handle 3 cases.

1) $\mathsf{sk}_0 = \mathsf{sk}_1$ and $\mathsf{msg}_0 \neq \mathsf{msg}_1$: Then,

$$\|(H(\mathsf{msg}_0) - H(\mathsf{msg}_1)) \cdot \mathsf{sk}_0 + (\mathbf{e}_0 - \mathbf{e}_1)\| \leq \sqrt{2}\beta,$$

which can further be bounded by $\|H(\mathsf{msg}_0) - H(\mathsf{msg}_1)) \cdot \mathsf{sk}_0\| \leq 2\sqrt{2}\beta$ since $\mathbf{e}_0, \mathbf{e}_1 \in \mathcal{K}\backslash\{\mathbf{0}\}$ for $\mathcal{K} = [-\beta, \beta]^n \cap \mathbb{Z}^n$. Furthermore, $\mathsf{sk}_0 \in \mathcal{K}\backslash\{\mathbf{0}\}$. Suppose $\mathcal{B}$ is challenged to find an R-SIS solution to $[\mathbf{a}_1, \mathbf{a}_2]$, where $\mathbf{a}_2$ is invertible in $\mathcal{R}_q$ with non-negligible probability. We can safely assume that the adversary $\mathcal{A}$ makes calls to the hash oracle to any messages before outputting its guess. Suppose that the adversary makes at most $q_H$ calls, where $\mathcal{B}$ selects $i_1, i_2 \in \{1, \ldots, q_H\}$ such that $i_1 < i_2$, which denote the critical calls. $\mathcal{B}$ maintains an $H_1$-List consisting of tuples $(\mathsf{msg}, \mathbf{h})$. If an entry has been set $\mathcal{B}$, outputs $\mathbf{h}$. Otherwise, if $\mathcal{A}$ makes his $i_1$ or $i_2$ query to the hash oracle, $\mathcal{B}$ samples a uniform random $\mathbf{r} \hookleftarrow \mathcal{R}_q$ and adds the entries $(\mathsf{msg}_0, -\mathbf{a}_2^{-1}\mathbf{r})$ and $(\mathsf{msg}_1, \mathbf{a}_2^{-1}(\mathbf{r} + \mathbf{a}_1))$ at positions $i_1$ and $i_2$ to the list. For all other queries $\mathcal{B}$ sets $(\mathsf{msg}, \mathbf{h})$ for a uniform random $\mathbf{h} \in \mathcal{R}_q$ and outputs $\mathbf{h}$. Eventually $\mathcal{A}$ outputs a valid $(\mathsf{msg}'_0, \tau_0, \mathsf{sk}_0, \mathsf{msg}'_1, \tau_1, \mathsf{sk}_1)$ such that $\mathsf{sk}_0 = \mathsf{sk}_1$ and $\|(H(\mathsf{msg}'_0) - H(\mathsf{msg}'_1)) \cdot \mathsf{sk}_0\| \leq 2\sqrt{2}\beta$. If neither $\mathsf{msg}'_0$ nor $\mathsf{msg}'_1$ have been queried at the $i_1$-th and $i_2$-th call, then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ computes $(\mathbf{a}_2^{-1}(\mathbf{r} + \mathbf{a}_1) - \mathbf{a}_2^{-1}\mathbf{r}) \cdot \mathsf{sk}_0 = \mathbf{s} \bmod q$, which is equivalent to

$$\mathbf{a}_1 \cdot \mathsf{sk}_0 + \mathbf{a}_2 \cdot \mathbf{s} \equiv \mathbf{0} \bmod q$$

and hence a solution to Ring-SIS$_{n,2,2\sqrt{2}\beta}$. Finally, $\mathcal{B}$ responds to the challenger with $[\mathsf{sk}_0, \mathbf{s}]$.

2) $\mathsf{sk}_0 \neq \mathsf{sk}_1$ and $\mathsf{msg}_0 = \mathsf{msg}_1$: Then,

$$\|H(\mathsf{msg}_0) \cdot (\mathsf{sk}_0 - \mathsf{sk}_1)\| \leq 2\sqrt{2}\beta,$$

with the relaxation $\|\mathsf{sk}_0 - \mathsf{sk}_1\| \leq \sqrt{2}\beta$. The proof is straight forward similar to the first case, however $\mathcal{B}$ needs only to program the oracle on the critical query at $i_1$ on $\mathsf{msg}_0$. In this case the $i_1$-th entry in the $H_1$-List is set to $(\mathsf{msg}_0, \mathbf{a}_2^{-1}\mathbf{a}_1)$. All other proof steps remain exactly the same.

3) $\mathsf{sk}_0 \neq \mathsf{sk}_1$ and $\mathsf{msg}_0 \neq \mathsf{msg}_1$: In this case, we have to solve Ring-SIS instance in a higher dimension:

$$\|H(\mathsf{msg}_0) \cdot \mathsf{sk}_0 - H(\mathsf{msg}_1) \cdot \mathsf{sk}_1\| \leq \sqrt{2}\beta\,.$$

The input problem instance to $\mathcal{B}$ is $[\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ for which he is challenged to output an SIS solution such that

$$\mathbf{a}_1 \cdot \mathbf{s}_1 + \mathbf{a}_2 \cdot \mathbf{s}_2 + \mathbf{a}_3 \cdot \mathbf{s}_3 \equiv \mathbf{0} \bmod q$$

and $\|\mathbf{s}_i\| \leq 2\sqrt{2}\beta$. Assuming that $\mathbf{a}_3$ is invertible, we proceed as in the previous cases with the small difference that the entries $(\mathsf{msg}_0, \mathbf{a}_3^{-1}\mathbf{a}_1)$ and $(\mathsf{msg}_1, \mathbf{a}_3^{-1}\mathbf{a}_2)$ are added to the $H_1$-list for the critical queries $i_1$ and $i_2$. All other steps are straight forward. This solves Ring-SIS$_{n,3,2\sqrt{2}\beta}$. □

## IV. Security Model

We adopt the most recent security model for DAA given by Camenisch et al. in [13]. The security definition is given in the Universal Composability (UC) framework [14] w.r.t. an ideal functionality $\mathcal{F}_{daa}^{\ell}$. In a UC model, an environment $\mathcal{E}$ should not be able to distinguish with more than negligible probability between two worlds; the first is the *real world* in which each party in the DAA protocol $\Pi$ executes its prescribed part of the protocol, in the presence of an adversary $\mathcal{A}$ that controls the network. The second is the *ideal world* in which all parties forward their inputs to a trusted third party, called the ideal functionality $\mathcal{F}_{daa}^{\ell}$, that performs all the required tasks for them, and sends back the desirable results. At a very high level, we want to show that the "real world" is *as correct and private as* the "ideal world". For correctness, we want to show that the outputs observed by the environment $\mathcal{E}$, while interacting with both worlds are the same; whereas for privacy, we want to show that whatever information the adversary $\mathcal{A}$ can learn in the real world, there exists a simulator $\mathcal{S}$ that can learn exactly the same information in the ideal world, so that the environment $\mathcal{E}$ can't tell if it is interacting with $(\mathcal{F}_{daa}^{\ell}, \mathcal{S})$ or $(\Pi, \mathcal{A})$. This simply shows that whatever damage the adversary $\mathcal{A}$ can do to the real world, there exists a simulator $\mathcal{S}$ that can do the same damage to the ideal world. In other words, whatever $\mathcal{A}$ can compute, can be computed given *only* the prescribed outputs. In a nutshell, the security properties that a DAA scheme should enjoy are the following:

**Unforgeability**. This property requires that the issuer is honest and should hold even if the host is corrupt. It guarantees that, if all the TPMs are honest, then no adversary can output a signature on a message msg w.r.t. a basename bsn. Moreover, if not all TPMs are honest, say $n$ TPMs are corrupt, then the adversary can *at most* output $n$ unlinkable signatures w.r.t. the same $\mathsf{bsn} \neq \perp$.

**Anonymity**. This property requires that the entire platform $(\mathsf{tpm}_i + \mathsf{host}_j)$ is honest and should hold even if the issuer is corrupt. It guarantees that, given two valid signatures w.r.t. two different bsn or $\mathsf{bsn} = \perp$, the adversary can't tell whether these signatures were produced by one or two different honest platforms.

**Non-frameability**. This requires that the entire platform $(\mathsf{tpm}_i + \mathsf{host}_j)$ is honest and should hold even if the issuer is corrupt. It guarantees that no adversary can produce a signature that links to signatures generated by an honest platform.

*Definition 4.1 (Universally Composable Security):* Given a protocol $\Pi$, an ideal functionality $\mathcal{F}_{daa}^{\ell}$ and an environment $\mathcal{E}$, we say that $\Pi$ securely realises $\mathcal{F}_{daa}^{\ell}$ if the real world in which $\Pi$ is used is as secure as the ideal world in which $\mathcal{F}_{daa}^{\ell}$ is used. More formally, for any adversary $\mathcal{A}$ in the real

world, there exists a simulator $\mathcal{S}$ in the ideal world such that $(\mathcal{E}, \mathcal{F}_{daa}^{\ell}, \mathcal{S}) \approx (\mathcal{E}, \Pi, \mathcal{A})$.

### A. The ideal Functionality $\mathcal{F}_{daa}^{\ell}$

Similar to [13], we will define our ideal functionality in the static setting, i.e. the adversary $\mathcal{A}$ decides which parties are corrupt and informs $\mathcal{F}_{daa}^{\ell}$ about them beforehand. $\mathcal{F}_{daa}^{\ell}$ has five interfaces, i.e. (Setup, Join, Sign, Verify, Link) which will be described below. In the UC model, several sessions of the protocol are allowed to run at the same time. Therefore, each session will be given a global identifier that consists of some issuer $\mathcal{I}$ and a unique string $\mathsf{sid}'$, i.e. $\mathsf{sid} = (\mathcal{I}, \mathsf{sid}')$; similarly, we will identify the Join and Sign sub-sessions by jsid and ssid. The parameter $\ell$ in $\mathcal{F}_{daa}^{\ell}$ represents a leakage function $\ell : \{0,1\}^* \to \{0,1\}^*$. This function $\ell$ models the leakage of information in the communication between a host $\mathsf{host}_j$ and a $\mathsf{tpm}_i$.

We define two useful functions that check whether or not a TPM key is consistent with the records of the ideal functionality $\mathcal{F}_{daa}^{\ell}$. We distinguish between the cases where the TPM is honest or corrupt as follows:

- $\mathsf{CheckGskHonestTPM}(\mathsf{gsk})$: if $\mathsf{tpm}_i$ is honest, and no signatures in Signed or valid signatures in VerResults identify to be signed by gsk, then gsk is eligible, and therefore return 1, otherwise return 0.
- $\mathsf{CheckGskCorruptTPM}(\mathsf{gsk})$: if $\mathsf{tpm}_i$ is corrupt and $\nexists \mathsf{gsk}' \neq \mathsf{gsk}$ and $(\mathsf{msg}, \sigma, \mathsf{bsn})$ s.t. both keys identify as the owners of this signature, then gsk is eligible, and therefore return 1, otherwise return 0

We also define the algorithms that will be used inside the functionality as follows:

- $\mathsf{kgen}(1^{\lambda})$: A probabilistic algorithm that generates keys for honest TPMs.
- $\mathsf{sig}(\mathsf{gsk}, \mathsf{msg}, \mathsf{bsn})$: A probabilistic algorithm that generates signatures for honest TPMs. On input a secret key gsk, a message msg, and a basename bsn, it outputs a signature $\sigma$.
- $\mathsf{ver}(\sigma, \mathsf{msg}, \mathsf{bsn})$: A deterministic algorithm that will be used in the Setup interface, which outputs 1 if $\sigma$ is a valid signature on msg w.r.t. bsn, and 0 otherwise.
- $\mathsf{link}(\sigma, \mathsf{msg}, \sigma', \mathsf{msg}', \mathsf{bsn})$: A deterministic algorithm that will be used in the Link interface, which outputs 1 if both $\sigma$ and $\sigma'$ were generated by the same TPM, and 0 otherwise.
- $\mathsf{identify}(\mathsf{gsk}, \sigma, \mathsf{msg}, \mathsf{bsn})$: A deterministic algorithm that will be used in various places to enforce consistency with the ideal functionality $\mathcal{F}_{daa}^{\ell}$'s internal records. It outputs 1 if gsk was used to produce $\sigma$, and 0 otherwise.

Note that the ideal functionality $\mathcal{F}_{daa}^{\ell}$ will use its internal records to enforce the correctness of the verification and linkage of the signatures it produces. However, ver() and link()'s role will be to help with signatures that weren't produced by the ideal functionality. Moreover, we require the link algorithm to be symmetric, i.e. $\mathsf{link}(\sigma, \mathsf{msg}, \sigma', \mathsf{msg}', \mathsf{bsn}) = \mathsf{link}(\sigma', \mathsf{msg}', \sigma, \mathsf{msg}, \mathsf{bsn})$, for all inputs. We now present the interfaces of $\mathcal{F}_{daa}^{\ell}$ as follows:

**Setup**. On input $(\mathsf{SETUP}, \mathsf{sid})$ from an issuer $\mathcal{I}$, $\mathcal{F}_{daa}^{\ell}$ does the following:

- Verify that $\mathsf{sid} = (\mathcal{I}, \mathsf{sid}')$ and output $(\mathsf{SETUP}, \mathsf{sid})$ to $\mathcal{S}$.
- Upon receiving the algorithms $(\mathsf{kgen}, \mathsf{sig}, \mathsf{ver}, \mathsf{link}, \mathsf{identify})$ from the simulator $\mathcal{S}$, it checks that $(\mathsf{ver}, \mathsf{link}, \mathsf{identify})$ are deterministic [Check−I].
- Output $(\mathsf{SETUPDONE}, \mathsf{sid})$ to $\mathcal{I}$.

**Join**. On input $(\mathsf{JOIN}, \mathsf{sid}, \mathsf{jsid}, \mathsf{tpm}_i)$ as a join request from a $\mathsf{host}_j$, that asks for the TPM $\mathsf{tpm}_i$ to join, the ideal functionality $\mathcal{F}_{daa}^{\ell}$ deals with it as follows:

JOIN REQUEST. Output a $(\mathsf{JOINSTART}, \mathsf{sid}, \mathsf{jsid}, \mathsf{tpm}_i, \mathsf{host}_j)$ to $\mathcal{S}$.

- Proceed upon receiving a delivery notification from $\mathcal{S}$.
- if $\mathcal{I}$ or $\mathsf{tpm}_i$ is honest and $< \mathsf{tpm}_i, -, - >$ is already in Members, output $\perp$ [Check−II].
- Otherwise, output a $(\mathsf{JOINPROCEED}, \mathsf{sid}, \mathsf{jsid}, \mathsf{tpm}_i)$ request to $\mathcal{I}$.

JOIN PROCEED. Proceed upon receiving an approval from $\mathcal{I}$, i.e. receiving $(\mathsf{JOINPROCEED}, \mathsf{sid}, \mathsf{jsid})$ from $\mathcal{I}$.

- Output $(\mathsf{JOINCOMPLETE}, \mathsf{sid}, \mathsf{jsid})$ to $\mathcal{S}$.
- On input $(\mathsf{JOINCOMPLETE}, \mathsf{sid}, \mathsf{jsid}, \mathsf{gsk})$ from $\mathcal{S}$, if both $(\mathsf{host}_j, \mathsf{tpm}_i)$ are honest:
  - Ignore the secret key gsk provided by the adversary, i.e. set $\mathsf{gsk} \leftarrow \perp$,
  - $\mathsf{Members.add}(< \mathsf{tpm}_i, \mathsf{host}_j, \mathsf{gsk} >)$
- If either of $(\mathsf{host}_j, \mathsf{tpm}_i)$ is not honest, it distinguishes between two cases to check if the provided gsk is an eligible one:
  - If $\mathsf{host}_j$ is corrupt and $\mathsf{tpm}_i$ is honest, $b \leftarrow \mathsf{CheckGskHonestTPM}(\mathsf{gsk})$ [Check−III].
  - Otherwise, if $\mathsf{tpm}_i$ is corrupt, $b \leftarrow \mathsf{CheckGskCorruptTPM}(\mathsf{gsk})$ [Check−IV].
  - If gsk is eligible, i.e. $b = 1$, then do $\mathsf{Members.add}(< \mathsf{tpm}_i, \mathsf{host}_j, \mathsf{gsk} >)$.
- Output $(\mathsf{JOINED}, \mathsf{sid}, \mathsf{jsid})$ to $\mathsf{host}_j$.

**Sign**. On input $(\mathsf{SIGN}, \mathsf{tpm}_i, \mathsf{msg}, \mathsf{bsn}, \sigma)$ from a host $\mathsf{host}_j$, that asks for a DAA signature by $\mathsf{tpm}_i$ on a message msg w.r.t. a basename bsn, the ideal functionality $\mathcal{F}_{daa}^{\ell}$ deals with the request as follows:

SIGN REQUEST. If the issuer $\mathcal{I}$ is honest and $< \mathsf{tpm}_i, \mathsf{host}_j, - > \notin$ Members, abort.

- Output $(\mathsf{SIGNSTART}, \mathsf{sid}, \mathsf{ssid}, \ell(\mathsf{msg}, \mathsf{bsn}), \mathsf{tpm}_i, \mathsf{host}_j)$ to $\mathcal{S}$.
- Proceed upon receiving a delivery notification from $\mathcal{S}$
- Output $(\mathsf{SIGNPROCEED}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$ to $\mathsf{tpm}_i$.

SIGN PROCEED. Proceed upon reception of $(\mathsf{SIGNPROCEED}, \mathsf{sid}, \mathsf{ssid})$ from $\mathsf{tpm}_i$, i.e. $\mathsf{tpm}_i$ approves.

- Output $(\mathsf{SIGNCOMPLETE}, \mathsf{sid}, \mathsf{ssid})$ to $\mathcal{S}$
- On input $(\mathsf{SIGNCOMPLETE}, \mathsf{sid}, \mathsf{ssid}, \sigma)$ from $\mathcal{S}$, if both the $\mathsf{host}_j$ and $\mathsf{tpm}_i$ are honest, then;
  - Ignore the signature $\sigma$ provided by the adversary,
  - If $\mathsf{bsn} \neq \perp$, look up $< \mathsf{tpm}_i, \mathsf{bsn}, \mathsf{gsk} >$ in DomainKeys and retrieve gsk. Else If $\mathsf{bsn} = \perp$ or no gsk was found, generate a fresh key $\mathsf{gsk} \leftarrow \mathsf{kgen}(1^{\lambda})$. If

$\mathsf{CheckGskHonestTPM}(\mathsf{gsk}) = 1$ [Check−V] , then add the new key to DomainKeys, i.e. $\mathsf{DomainKeys.add}(< \mathsf{tpm}_i, \mathsf{bsn}, \mathsf{gsk} >)$.
  - Generate $\sigma \leftarrow \mathsf{sig}(\mathsf{gsk}, \mathsf{msg}, \mathsf{bsn})$ and check that $\mathsf{ver}(\sigma, \mathsf{msg}, \mathsf{bsn}) = 1$ [Check−VI].
  - Check that $\mathsf{identify}(\sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{gsk}) = 1$ [Check−VII].
  - Check that no $\mathsf{tpm}'_i \neq \mathsf{tpm}_i$ exists such that $(\mathsf{tpm}'_i, \mathsf{gsk}')$ is in Members or DomainKeys with $\mathsf{identify}(\sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{gsk}') = 1$ [Check−VIII].
- If $\mathsf{tpm}_i$ is honest, then do $\mathsf{Signed.add}(< \sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{tpm}_i >)$.
- Output $(\mathsf{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, \sigma)$ to $\mathsf{host}_j$.

**Verify**. This interface takes as input $(\mathsf{VERIFY}, \mathsf{sid}, \sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{RL})$, it can be called by any party $\mathcal{V}$ to check whether or not a given signature $\sigma$ is a valid signature on a message msg w.r.t. the basename bsn and the revocation list RL; the ideal functionality deals with this request as follows:

- Extract all pairs $(\mathsf{gsk}_i, \mathsf{tpm}_i)$ from DomainKeys and Members, for which $\mathsf{identify}(\sigma, \mathsf{msg}, \mathsf{bsn}, gsk) = 1$. Set $b = 0$ If any of the following conditions holds:
  - Distinct keys $\mathsf{gsk}_i$s were found [Check−IX].
  - $\mathcal{I}$ is honest and no $(\mathsf{gsk}_i, \mathsf{tpm}_i)$ was found [Check−X].
  - a $\mathsf{gsk}_i$ was found, but no entry of the form $(< *, \mathsf{msg}, \mathsf{bsn}, \mathsf{tpm}_i >)$ was found in Signed [Check−XI].
  - There exists a $\mathsf{gsk}' \in \mathsf{RL}$ for which $\mathsf{identify}(\sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{gsk}') = 1$ and no $(\mathsf{gsk}_i, \mathsf{tpm}_i)$ for an honest $\mathsf{tpm}_i$ was found [Check−XII].
- If $b \neq 0$, set $b \leftarrow \mathsf{ver}(\sigma, \mathsf{msg}, \mathsf{bsn})$ [Check−XIII].
- $\mathsf{VerResults.add}(< \sigma, \mathsf{msg}, \mathsf{bsn}, \mathsf{RL}, b >)$.
- Output $(\mathsf{Verified}, \mathsf{sid}, b)$ to $\mathcal{V}$

**Link**. This interface takes as input $(\mathsf{LINK}, \mathsf{sid}, \sigma_1, \mathsf{msg}_1, \sigma_2, \mathsf{msg}_2, \mathsf{bsn})$ and can be called by any party $\mathcal{V}$ to check whether or not two given signatures that are directed at the same recipient $\mathsf{bsn} \neq \perp$ stem from the same signer. The ideal functionality deals with this request as follows:

- If any of $(\sigma_i, \mathsf{msg}_i, \mathsf{bsn}, \mathsf{RL} = \emptyset) = 0$ for $i = 1$ or 2 is not valid (using the Verify interface with $\mathsf{RL} = \emptyset$), output $\perp$ to $\mathcal{V}$ [Check−XIV].
- Compute $a_{ij} \leftarrow \mathsf{identify}(\sigma_i, \mathsf{msg}_i, \mathsf{bsn}, \mathsf{gsk}_j)$, for all $\mathsf{gsk}_j$ in Members and DomainKeys, for $i = 1, 2$
  - If there exists a $\mathsf{gsk}_j$ s.t. $a_{1j} \neq a_{2j}$, set $b = 0$ [Check−XV].
  - Else if there exists a $\mathsf{gsk}_j$ s.t. $a_{1j} = a_{2j} = 1$, set $b = 1$ [Check−XVI].
- If $b$ is not defined yet, set $b \leftarrow \mathsf{link}(\sigma_1, \mathsf{msg}_1, \sigma_2, \mathsf{msg}_2, \mathsf{bsn})$
- Output $(\mathsf{LINK}, \mathsf{sid}, b)$ to $\mathcal{V}$.

### B. $\mathcal{F}^{\ell}_{daa}$'s Enforcement of DAA's Security Properties

We now show that $\mathcal{F}^{\ell}_{daa}$ indeed enforces the desired security properties of DAA, i.e. unforgeability, anonymity, and non-frameability. We refer to [13] for a more detailed discussion, in particular, to show the consistency of verify, and the consistency and symmetry of link.

**Completness**. A signature $\sigma$ produced by an honest $\mathsf{tpm}_i$ and honest $\mathsf{host}_j$ on message msg and w.r.t. a basename bsn, should be accepted by the verifier. The ideal functionality $\mathcal{F}^{\ell}_{daa}$ ensures this property and this can be easily checked by looking at the internal checks [Check−IX, X, XI, XII].

**Unforgeablity**. For the first unforgeability scenario, i.e. when all TPMs are honest, the adversary shouldn't be able to produce a signature on a message msg w.r.t. basename bsn if no honest TPM signed msg w.r.t. to a basename bsn. This is indeed ensured by [Check−X] and [Check−XI]. The former ensures that the signature must trace back to some TPM's gsk whereas the latter ensures that any signature on messages not signed by that TPM will get rejected. For the second forging scenario, i.e. some TPMs are corrupt (say $n$ corrupt ones), the adversary shouldn't be able to produce more than $n$ non-linkable signatures w.r.t. the same $\mathsf{bsn} \neq \perp$. Assume that the adversary manages to output $n + 1$ signatures, one can easily show, by the internal checks [Check−X, XI, XVI], that this gives a contradiction, i.e. these $n + 1$ signatures can't simultaneously verify w.r.t. same non-empty bsn and be pairwise unlinkable.

**Anonymity**. This ensures that signatures produced for honest platforms $(\mathsf{tpm}_i + \mathsf{host}_j)$ by $\mathcal{F}^{\ell}_{daa}$ are always anonymous. This is due to the fact that $\mathcal{F}^{\ell}_{daa}$ always uses fresh keys to sign messages w.r.t. different basenames or to empty ones. Therefore, the distribution of the signatures with different or empty basenames is independent of the platforms. On the other hand, [Check−XII] makes sure that exploit the revocation property to break the anonymity of honest users, i.e. $\mathcal{F}^{\ell}_{daa}$ ignores the $\{\mathsf{gsk}_i\} \in \mathsf{RL}$ that belong to honest TPMs when testing for revocation. Note that the simulator is allowed to provide the signatures for corrupt platforms, which implies that anonymity can't be guaranteed here as those signatures might depend on the identity of the signers.

**Non-frameability**. An adversary should not be able to frame honest platforms, i.e he shouldn't be able to produce signatures on messages that were never signed by the platform and yet link to signatures signed by this platform. We stress here that this property can't be achieved if only the TPM is honest but the host is corrupt. This property is indeed guaranteed by the internal checks [Check−XIV, XI, XV].

## V. Our Construction: A (stateful) Lattice-based Direct Anonymous Attestation (DAA) Protocol

Following [13] we use the following abstractions and assume that there exist secure realizations of some standard functionalities. First, we denote by $\mathcal{F}_{\mathsf{ca}}$ a common certificate authority functionality that is available to all parties. A common reference ring $\mathcal{F}_{\mathsf{crs}}$ provides the participants with all the system parameters. These functionalities are accessed by parties if key material of other participants are needed. Furthermore, it is required to have an authenticated channel between the issuer and the TPM in the Join protocol, where the host has the power to block the communication. Since the authentication mechanism can be realized in different ways,

this feature is captured by the ideal functionality $\mathcal{F}_{\mathsf{auth}^*}$. For a secure message transmission between the TPM and the host we exploit the functionality $\mathcal{F}_{\mathsf{smt}}$, which is normally assumed to be given due to the close physical proximity of both participants on the platform.

1) Setup: The issuer $\mathcal{I}$ creates a key-pair for Boyen's signature scheme. Subsequently he registers the key with $\mathcal{F}_{\mathsf{ca}}$.
   Upon input $(\mathsf{SETUP}, \mathsf{sid})$ the $\mathcal{I}$ proceeds as follows to generate his key pair:
   - He checks that $sid = (\mathcal{I}, \mathsf{sid}')$.
   - He invokes $\mathsf{Gen}(1^n)$ of the Boyen signature scheme in combination with the trapdoor construction [26], [4] in order to obtain a trapdoor $\hat{\mathbf{R}} = [\hat{\mathbf{R}}_1, \ldots, \hat{\mathbf{R}}_k]$ with small entries and a verification key $(\hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u})$, where $\hat{\mathbf{A}}_i$ defines a uniform random vector of polynomials and $\hat{\mathbf{A}} = [\hat{\mathbf{B}}, \mathbf{g}_1 - h_{\hat{\mathbf{B}}}(\hat{\mathbf{R}}_1), \ldots, \mathbf{g}_k - h_{\hat{\mathbf{B}}}(\hat{\mathbf{R}}_k)\ ] \in \mathcal{R}_q^m$ for $k = \lceil \log q \rceil, m = 2k, \mathbf{g}_i = 2^{i-1}$ and a generalized knapsack function $h_{\hat{\mathbf{B}}}(\hat{\mathbf{C}}) = \sum_i \mathbf{b}_i \cdot \mathbf{c}_i$ with $\hat{\mathbf{B}} = [\mathbf{b}_1, \ldots, \mathbf{b}_k]$ and $\hat{\mathbf{C}} = [\mathbf{c}_1, \ldots, \mathbf{c}_k]$. He further generates a random polynomial $\mathbf{b} \hookleftarrow \mathcal{R}_q$ and sets $\mathsf{isk} := \hat{\mathbf{R}}$ and $\mathsf{gpk} := (\mathbf{b}, \hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u})$.
   The issuer $\mathcal{I}$ produces a signature of knowledge for the correct generation of the verification keys, i.e.

$$\begin{aligned}\pi_0 \quad = \quad & \mathsf{SPK}\{ \\ & \mathsf{public} := \{\mathbf{b}, \hat{\mathbf{A}}, \{\hat{\mathbf{A}}_i\}_{i=1}^{\ell}, \mathbf{u}\}, \mathsf{witness} := \{\hat{\mathbf{R}}\} : \\ & \hat{\mathbf{A}} = [\hat{\mathbf{B}}, \mathbf{g}_1 - h_{\hat{\mathbf{B}}}(\hat{\mathbf{R}}_1), \ldots, \mathbf{g}_k - h_{\hat{\mathbf{B}}}(\hat{\mathbf{R}}_k)\ ] \wedge \\ & \|\ \hat{\mathbf{R}}_i\ \| \leq \beta \text{ for } 1 \leq i \leq k\}\end{aligned}$$

   This proof can be generated in different ways such as the protocol from Section V-B or standard protocols for small SIS secrets. Finally, the issuer publishes the public key $\mathsf{gpk} := (\mathbf{b}, \hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u})$ together with the proof $\pi_0$.
   - He initializes the list of joining participants $\mathsf{Members} \leftarrow \emptyset$.
   - Finally, he registers $(\pi_0, \mathsf{gpk})$ at $\mathcal{F}_{\mathsf{ca}}$, stores its private key isk and outputs $(\mathsf{SETUPDONE}, \mathsf{sid})$.

2) Join Request: Assuming the correctness of the public key gpk via its associated proof, a secure communication channel between $\mathcal{P}_i$ and $\mathcal{I}$ is established using for instance the TPM endorsement secret key. The issuer $\mathcal{I}$ verifies that the certificate of the endorsement public key of the TPM is valid and checks if the TPM-host tuple is qualified to receive an attestation. In case of acceptance, $\mathcal{I}$ outputs a credential enabling $\mathcal{P}_i$ to create attestations. Via the unique sub-session identifier jsid, $\mathcal{I}$ can differentiate various join sessions that are executed simultaneously.
   - On input query $(\mathsf{JOIN}, \mathsf{sid}, \mathsf{jsid}, \mathsf{tpm}_i)$, where sid is parsed as $\mathsf{sid} = (\mathcal{I}, \mathsf{sid}')$ the host $\mathsf{host}_j$ forwards $(\mathsf{JOIN}, \mathsf{sid}, \mathsf{jsid})$ to $\mathcal{I}$.
   - On input query $(\mathsf{JOIN}, \mathsf{sid}, \mathsf{jsid})$ sent by $\mathsf{host}_j$ the issuer $\mathcal{I}$ replies by $(\mathsf{sid}, \mathsf{jsid}, \mu)$ back to $\mathsf{host}_j$ for a uniform random nonce $\mu \leftarrow_R \{0,1\}^\lambda$.
   - The message $(\mathsf{sid}, \mathsf{jsid}, \mu)$ sent by $\mathcal{I}$ to $\mathsf{host}_j$ is subsequently forwarded to $\mathsf{tpm}_i$.
   - When $\mathsf{tpm}_i$ receives $(\mathsf{sid}, \mathsf{jsid}, \mu)$, it proceeds as follows
     i) It first checks that no such entry exists in it storage.
     ii) Subsequently, it samples secret polynomials
     $$\hat{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_{2m+1}) \hookleftarrow D_{\mathbb{Z}^n, s_1} \times D_{\mathbb{Z}^n, s_2} \times \ldots \times D_{\mathbb{Z}^n, s_2}$$
     and stores its key as $(\mathsf{sid}, \mathsf{host}_j, \hat{\mathbf{y}}, \perp)$.
     iii) The TMP $\mathsf{tpm}_i$ computes $\tilde{\mathbf{u}} = [\ \mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}\ ] \cdot \hat{\mathbf{y}} \bmod q$ for $\mathsf{id} = i$, which statistically hides $\mathbf{y}_1$, and generates
     $$\begin{aligned}\pi_1 \quad = \quad & \mathsf{SPK}\{\mathsf{public} := \{\mathsf{bsn}_{\mathcal{I}}, \mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}, \tilde{\mathbf{u}}\}, \\ & \mathsf{witness} := \{\hat{\mathbf{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_{2m+1})\} : \\ & \{\tilde{\mathbf{u}} = [\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \hat{\mathbf{y}} \bmod q \wedge \|\hat{\mathbf{y}}\| \leq \beta_1\} \\ \wedge \quad & \{\mathsf{nym} = H(\mathsf{bsn}_{\mathcal{I}}) \cdot \mathbf{y}_1 + \mathbf{e} \wedge \|\mathbf{e}\| \leq \beta_2\}\}(n)\,.\end{aligned}$$
     iv) It stores the record $(\mathsf{sid}, \mathsf{host}_j, \hat{\mathbf{y}}, \mathsf{id})$.
     v) Finally, the message $(\mathsf{nym}, \tilde{\mathbf{u}}, \mathsf{id}, \pi_1)$ is sent via $\mathsf{host}_j$ to $\mathcal{I}$ by means of $\mathcal{F}_{\mathsf{auth}^*}$.
   - The host extends the unauthenticated part of the message by appending its identity to it and subsequently forwards the complete message to $\mathcal{I}$ and keeps the state $(\mathsf{jsid}, \tilde{\mathbf{u}}, \mathsf{id})$.
   - The issuer receives an authenticated message $(\mathsf{nym}, \tilde{\mathbf{u}}, \mathsf{id}, \pi_1)$ from $\mathsf{tpm}_i$ together with the unauthenticated identity of $\mathsf{host}_j$. It then verifies the proof $\pi_1$ and makes sure that $\mathsf{tpm}_i \notin \mathsf{Members}$. It stores $(\mathsf{jsid}, \mathsf{nym}, \tilde{\mathbf{u}}, \mathsf{id}, \mathsf{tpm}_i, \mathsf{host}_j)$ and subsequently generates the message $(\mathsf{JOINPROCEED}, \mathsf{sid}, \mathsf{jsid}, \mathsf{tpm}_i)$.

3) Join Proceed: Once the platform wishes to proceed with the join session, the message $(\mathsf{JOINPROCEED}, \mathsf{sid}, \mathsf{jsid})$ is sent over to the issuer, which in turn generates a valid credential:
   - Upon input $(\mathsf{JOINPROCEED}, \mathsf{sid}, \mathsf{jsid})$ the issuer proceeds as follows:
     i) First, the issuer looks up the record $(\mathsf{jsid}, \mathsf{nym}, \tilde{\mathbf{u}}, \mathsf{id}, \mathsf{tpm}_i, \mathsf{host}_j)$ and puts $\mathsf{tpm}_i$ into the list Members.
     ii) The issuer samples a preimage $\hat{\mathbf{x}} \in \mathcal{R}^{2m}$ of $\mathbf{u} - \tilde{\mathbf{u}}$ as a credential for the vector of polynomials $\hat{\mathbf{A}}_{\mathsf{id}}$.
     iii) The message $(\mathsf{sid}, \mathsf{jsid}, \hat{\mathbf{x}})$ is subsequently sent to $\mathsf{host}_j$ via $\mathcal{F}_{\mathsf{smt}}$
   - When $\mathsf{host}_j$ receives the message $(\mathsf{sid}, \mathsf{jsid}, \hat{\mathbf{x}})$, it proceeds as follows:
     i) It checks that $\hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{x}} = \mathbf{u} - \tilde{\mathbf{u}} \bmod q$ and $\|\hat{\mathbf{x}}\| \leq \beta$.
     ii) If satisfied, it stores $(\mathsf{sid}, \mathsf{tpm}_i, \tilde{\mathbf{u}}, \mathsf{id}, \hat{\mathbf{x}})$ and then outputs $(\mathsf{JOINED}, \mathsf{sid}, \mathsf{jsid})$.

4) Sign Request: Once the join protocol is completed, $\mathsf{tpm}_i$ and $\mathsf{host}_j$ can interactively sign a message msg under basename bsn. In order to distinguish different sub-sessions, a unique identifier ssid is used.
   - Upon input $(\mathsf{SIGN}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$ the host proceeds as follows:

i) It looks up the record $(\mathsf{sid}, \mathsf{tpm}_i, \tilde{\mathbf{u}}, \mathsf{id}, \hat{\mathbf{x}})$.
ii) The message $(\mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$ is subsequently sent to $\mathsf{tpm}_i$.

- When $\mathsf{tpm}_i$ receives the message $(\mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$, it proceeds as follows:
  i) It makes sure to have a join record $(\mathsf{sid}, \mathsf{host}_j, \hat{\mathbf{y}}, \mathsf{id})$.
  ii) Subsequently, it generates a sign entry $(\mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$ in its storage and outputs $(\mathsf{SIGNPROCEED}, \mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$.

5) Sign Proceed: After $\mathsf{host}_j$ grants permission to proceed, the signature is generated.

- Upon input $(\mathsf{SIGNPROCEED}, \mathsf{sid}, \mathsf{ssid})$ the TPM proceeds as follows:
  i) It retrieves the join record $(\mathsf{sid}, \mathsf{host}_j, \hat{\mathbf{y}}, \mathsf{id})$ and sign record $(\mathsf{sid}, \mathsf{ssid}, \mathsf{msg}, \mathsf{bsn})$.
  ii) In order to randomize its signature, $\mathsf{tpm}_i$ samples a uniform random vector of polynomials $\hat{\mathbf{t}}$ with integer coefficients in $[-d, d]$ such that
  $$\hat{\mathbf{A}}_{\mathsf{id}} \cdot (\hat{\mathbf{y}} + [\mathbf{0}, \hat{\mathbf{t}}]) = \mathbf{u}'$$
  is uniform random.
  iii) Depending on the input bsn, there are two cases to be considered:

  **Case** $\mathsf{bsn} = \perp$:
  The TPM samples $\mathbf{v} \hookleftarrow H(\mu)$ for a random seed $\mu \leftarrow \{0,1\}^\lambda$, computes $\mathsf{nym} = \mathbf{v} \cdot \mathbf{y}_1 + \mathbf{e} = \mathsf{TAG}_{\mathsf{LIT}}(\mu, \mathbf{y}_1)$ for an error term $\mathbf{e}$ such that $\|\mathbf{e}\| \leq \beta_2$ and generates a signature of knowledge

  $$\begin{aligned}\pi_2 \;=\; & \mathsf{SPK}\{ \\ & \mathsf{public} := \{\mathbf{v}, \mathbf{b}, \hat{\mathbf{A}}, \{\hat{\mathbf{A}}_i\}_{i=1}^{\ell}, \mathbf{u}, \mathsf{nym}, \mathsf{bsn}\}, \\ & \mathsf{witness} := \{\mathsf{sk} = \hat{\mathbf{y}} + [\mathbf{0}, \hat{\mathbf{t}}], \mathsf{id}\} : \\ & \{[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \mathsf{sk} = \mathbf{u}' \wedge \|\mathsf{sk}\| \leq \beta + d\} \\ & \wedge \{\mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mu, \mathbf{y}_1)\}\}(\mathsf{msg})\,.\end{aligned}$$

  Finally, it sends $(\pi_2, \hat{\mathbf{t}})$ to $\mathsf{host}_j$, where nym and $\mu$ are parts of $\pi_2$.

  **Case** $\mathsf{bsn} \neq \perp$:
  The TPM looks up in its storage for a basename-tag pair $(\mathsf{bsn}, \mathsf{nym})$ such that $\mathsf{LINK}_{\mathsf{LIT}}(\mathsf{TAG}_{\mathsf{LIT}}(\mathsf{bsn}, \mathbf{y}_1), \mathsf{nym}) = 1$, otherwise it generates a tag $\mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{bsn}, \mathbf{y}_1)$ and stores the respective entry. The TPM generates a signature of knowledge

  $$\begin{aligned}\pi_2 \;=\; & \mathsf{SPK}\{ \\ & \mathsf{public} := \{\mathbf{b}, \hat{\mathbf{A}}, \{\hat{\mathbf{A}}_i\}_{i=1}^{\ell}, \mathbf{u}, \mathsf{nym}, \mathsf{bsn}\}, \\ & \mathsf{witness} := \{\mathsf{sk} = \hat{\mathbf{y}} + [\mathbf{0}, \hat{\mathbf{t}}], \mathsf{id}\} : \\ & \{[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \mathsf{sk} = \mathbf{u}' \wedge \|\mathsf{sk}\| \leq \beta + d\} \\ & \wedge \mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{bsn}, \mathbf{y}_1)\}(\mathsf{msg}),\end{aligned}$$

  iv) Finally, it sends $(\mathsf{sid}, \mathsf{ssid}, \pi_2, \hat{\mathbf{t}}, \mathsf{msg})$ to $\mathsf{host}_j$, where nym and bsn are parts of $\pi_2$.

- When $\mathsf{host}_j$ receives the message $(\mathsf{sid}, \mathsf{ssid}, \pi_2, \hat{\mathbf{t}}, \mathsf{msg})$, it proceeds as follows:
  i) It checks that the proof $\pi_2$ is valid and $\|\hat{\mathbf{t}}\| \leq d$.
  ii) Subsequently, it generates a signature of knowledge for the same message msg

  $$\begin{aligned}\pi_3 \;=\; & \mathsf{SPK}\{ \\ & \mathsf{public} := \{\mathbf{b}, \hat{\mathbf{A}}, \{\mathbf{A}_i\}_{i=1}^{\ell}, \mathbf{u} - \mathbf{u}', \mathsf{msg}, \pi_2\}, \\ & \mathsf{witness} := \{\mathsf{sk} = [0, \hat{\mathbf{x}} - \hat{\mathbf{t}}], \mathsf{id}\} : \\ & \{\hat{\mathbf{A}}_{\mathsf{id}} \cdot \mathsf{sk} = \mathbf{u} - \mathbf{u}' \\ & \wedge \|\mathsf{sk}\| \leq \beta + d\}\}(\mathsf{msg}).\end{aligned}$$

  iii) Finally, the host outputs
  $$(\mathsf{SIGNATURE}, \mathsf{sid}, \mathsf{ssid}, \sigma = (\pi_2, \pi_3, \mathbf{u}', \mathsf{msg}))$$
  as the signature.

*Remark.* Both $\pi_2$ and $\pi_3$ prove the knowledge of short elements $\hat{\mathbf{z}}_1$ and $\hat{\mathbf{z}}_2$ such that $[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \hat{\mathbf{z}}_1 = \mathbf{u}'$ and $\hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{z}}_2 = \mathbf{u} - \mathbf{u}'$. Together they can combine their secret data to obtain a valid credential satisfying $[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}} \cdot (\hat{\mathbf{z}}_1 + [\mathbf{0}, \hat{\mathbf{z}}_2]) = \mathbf{u}$ with $\|\hat{\mathbf{z}}_1 + [\mathbf{0}, \hat{\mathbf{z}}_2]\| \leq 2(\beta + d)$. Since there are only polynomially many platforms involved, the probability that two parties from different platforms can combine their secret data to a valid credential system is negligible. Otherwise this would mean that two different parties have sampled polynomials mapping to the same syndrome, which is uniform random for discrete Gaussian distributed polynomials with parameter $s \geq \eta_\epsilon(\Lambda_q^\perp(\hat{\mathbf{A}}))$. We note that we can also distribute the generation of tags to both parties in case this procedure has to be shared among the TPM and the host. This technique may be of independent interest.

6) Verify: The verifier checks that the obtained signature $\sigma$ is valid given the message msg, basename bsn, private key revocation list RL.

- Upon input $(\mathsf{VERIFY}, \mathsf{sid}, \sigma, \mathsf{msg})$ the verifier $\mathcal{V}$ checks the signature
  i) He parses $\sigma$ as $\pi_2, \pi_3, \mathbf{u}', \mathsf{nym}, \mathsf{bsn}, \mu$
  ii) He checks the SPKs $\pi_2$ and $\pi_3$ with respect to $\mathsf{bsn}, \mu, \mathsf{nym}, \mathsf{msg}$ and $\mathbf{u}'$ or $\mathbf{u} - \mathbf{u}'$, which together represent a valid signature on msg with respect to a valid credential satisfying
  $$[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \hat{\mathbf{z}} = \mathbf{u} \wedge \|\hat{\mathbf{z}}\| \leq 2(\beta + d) \wedge \mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{str}, \mathbf{z}_1),$$
  where either $\mathsf{str} = \mathsf{bsn}$ or $\mathsf{str} = \mu$ holds for a uniform random string $\mu$.
  iii) He also checks that no key in $\mathsf{sk} = [\mathbf{y}_1, \ldots, \mathbf{y}_{2m+1}] \in \mathsf{RL}$ has been used to generate nym, i.e. $\mathsf{LINK}_{\mathsf{LIT}}(\mathsf{nym}, H(\mathsf{str}) \cdot \mathbf{y}_1) = 0$.
  iv) Finally, $\mathcal{V}$ outputs $(\mathsf{VERIFIED}, \mathsf{ssid}, \mathsf{out})$, where $\mathsf{out} = 1$ if all checks are successful, otherwise $\mathsf{out} = 0$.

7) Link: Direct anonymous attestation protocols allow the

user to generate signatures that can be linked to previously issued signatures. This is realized in our protocol via a tag that is appended to the signature and is generated from a basename bsn, which can take as values a random bit string or the pseudonym of the verifier's name. In the first case, the verifier is not able to link signatures of the same signer. In the second case, the verifier can link and identify all signatures of a particular signer/TPM, where the tag is generated from the same base name.

- The verifier $\mathcal{V}$ upon input $(\mathsf{LINK}, \mathsf{sid}, \sigma, \mathsf{msg}, \sigma', \mathsf{msg}', \mathsf{bsn})$ proceeds as follows:
  - i) The signatures $\sigma, \sigma'$ are parsed as $\sigma = \pi_2, \pi_3, \mathbf{u}_a, \mathsf{nym}, \mathsf{bsn}, \mu$ and $\sigma' = \pi_2', \pi_3', \mathbf{u}_b, \mathsf{nym}', \mathsf{bsn}', \mu'$, respectively. He outputs $\perp$ if the signatures $\sigma, \sigma'$ are not valid.
  - ii) If both are valid, the verifier checks that $\mathsf{bsn} = \mathsf{bsn}'$ and computes $\mathsf{out} = \mathsf{LINK}_{\mathsf{LIT}}(\mathsf{nym}, \mathsf{nym}') \in \{0,1\}$. In case of $\mathsf{out} = 1$ the same TPM signed $\sigma$ and $\sigma'$, otherwise two different signers were involved.
  - iii) Finally, the TPM outputs $(\mathsf{LINK}, \mathsf{sid}, \mathsf{out})$.

### A. Security

In the following section we show that our construction is secure in the model presented in Section IV, which is inspired by the work by Camenisch et al. in [13]. In particular, we instantiate the SPKs using Unruh's transformation [30] to achieve a non-interactive zero-knowledge proof system providing online extractability and security in the quantum random oracle model as well. But as noted in [13] one could in principal also use extractability by rewinding (e.g. via Fiat-Shamir), which puts forward a more efficient scheme. However, care has to be taken such that the security proof does not rely on rewinding consuming exponential time. This is avoided, for instance, by putting a constrain on the number of simultaneous sessions of the join protocol, where $\pi_1$ is subject to rewinding. Using a bound that is logarithmic in the security parameter, the scheme gets secure, however in a standalone fashion only and not in the UC model, which prohibits rewinding in order to claim composability.

*Theorem 5.1:* In the $(\mathcal{F}_{\mathsf{crs}}, \mathcal{F}_{\mathsf{ca}}, \mathcal{F}_{\mathsf{auth}^*}, \mathcal{F}_{\mathsf{smt}})-$hybrid model using random oracles and static corruptions the protocol $\Pi_{daa}$ given in Section V securely realizes $\mathcal{F}_{daa}^{\ell}$ assuming the hardness of (Ring-)LWE and (Ring-)SIS, the unforgeability of Boyen signatures and the MAC scheme, and the proofs-of-knowledge are online extractable.

### B. Zero-Knowledge Argument of Knowledge

In this section, we show how to adapt the sZKAoK provided in [24] to the ring setting in order to prove the language (cf. [24] for the matrix variant).

$$\begin{aligned}\pi \;=\; & \{\mathsf{public} := \{\mathbf{c}, \hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell, \mathbf{u}, \mathsf{nym}, \mu, \mathsf{msg}\},\\ & \mathsf{witness} := \{\mathsf{id}, \hat{\mathbf{z}} = (\mathbf{z}_0, \ldots, \mathbf{z}_{2m})\} :\\ & \{[\mathbf{c}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \hat{\mathbf{z}} = \mathbf{u} \wedge \|\hat{\mathbf{z}}\| \leq \beta\} \wedge\\ & \{\mathsf{nym} = H(\mathsf{msg}, \mu) \cdot \mathbf{z}_0 + \mathbf{e} \wedge \|\mathbf{e}\| \leq \beta\}\},\end{aligned}$$

where $\mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}((\mathsf{msg}, \mu), \mathbf{z}_0) = H(\mathsf{msg}, \mu) \cdot \mathbf{z}_0 + \mathbf{e}$. Since we are operating in the ring $\mathcal{R}_q$, we can transform any linear transformation into matrix vector products. We construct the matrices $\mathbf{B}_0 = \mathsf{rot}(\mathbf{b})$ and $\mathbf{B}_i = \mathsf{rot}(\mathbf{a}_i) \in \mathbb{Z}_q^{n \times n}$ for all polynomials in $\hat{\mathbf{A}}, \hat{\mathbf{A}}_0, \ldots, \hat{\mathbf{A}}_\ell$ leading to $\mathbf{B}_1, \ldots, \mathbf{B}_{(\ell+2)m}$. The error polynomial $\mathbf{e}$ and the polynomials in $\hat{\mathbf{z}} = (\mathbf{z}_0, \ldots, \mathbf{z}_{2m}) = (\mathbf{z}_0, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2)$ are now considered as vectors in $\mathbb{Z}^n$ such that

$$\mathbf{B}_0 \cdot \mathbf{z}_0 + \underbrace{\sum_{i=1}^{2m} \mathbf{B}_i \cdot \mathbf{z}_i}_{[\hat{\mathbf{A}} \mid \hat{\mathbf{A}}_0] \cdot \hat{\mathbf{z}}} + \sum_{j=1}^{\ell} \mathsf{id}_j \cdot \underbrace{\sum_{i=1}^{m} \mathbf{B}_{i+(j+2)m} \cdot \mathbf{z}_{i+m}}_{\hat{\mathbf{A}}_j \cdot \hat{\mathbf{z}}_2} = \mathbf{u} \bmod q\,.$$

Furthermore, id is extended to $\mathsf{id}^* \in \mathsf{B}_{2\ell}$, which is the set of vectors of hamming weight $\ell$. Using the Extension-Decomposition technique and letting $k = \lfloor \log \beta \rfloor + 1$, we can decompose and extend $\mathbf{e}$ and each vector $\mathbf{z}_i$ into $k$ vectors $\{\mathbf{e}^j\}_{j=1}^k$, $\{\mathbf{z}_0^j\}_{j=1}^k, \ldots, \{\mathbf{z}_{2m}^j\}_{j=1}^k \in \{-1, 0, 1\}^{3n}$ such that

$$\sum_{i=0}^{2m} \mathbf{B}_i^* \underbrace{(\sum_{d=1}^{k} \beta_d \mathbf{z}_i^d)}_{\mathbf{z}_i} + \sum_{j=1}^{2\ell} \mathsf{id}_j \sum_{i=1}^{m} \mathbf{B}_{i+(j+2)m}^* \underbrace{(\sum_{d=1}^{k} \beta_d \mathbf{z}_{i+m}^d)}_{\mathbf{z}_{i+m}} = \mathbf{u} \bmod q,$$

where all $\mathbf{z}_i^d$ contain exactly the same number of entries from each of $\{-1, 0, 1\}$ and $\{\beta_i\}_{i=1}^k$ represents a decomposition of the interval $[0, \beta]$ such that $|\sum_{i=1}^k \beta_i x_i| \leq \beta$ for any $x_i \in \{-1, 0, 1\}$. The extended matrices are of the following form $\mathbf{B}_i^* = [\mathbf{B}_i \mid \mathbf{0} \in \mathbb{Z}^{n \times 2n}]$ and $\mathbf{B}_{i+(j+2)m}^* = \mathbf{0}$ for $j > \ell$. Let therefore $\mathbf{z}_i = \mathbf{0} \in \mathbb{Z}^n$ for $2m < i \leq (2+2\ell)m$. A set of vectors $\mathbf{v} = [\mathbf{v}_0, \ldots, \mathbf{v}_{(2+2\ell)m}]$ is said to be valid with respect to $\mathsf{id}^* \in \mathsf{B}_{2\ell}$, if

$$\begin{aligned}\mathbf{v} \;=\; & [\ \mathbf{v}_0, \ldots, \mathbf{v}_{2m} \ ||\mathsf{id}_1^* \cdot (\mathbf{v}_{m+1}, \ldots, \mathbf{v}_{2m})||\\ & \ldots || \ \mathsf{id}_{2\ell}^* \cdot (\mathbf{v}_{m+1}, \ldots, \mathbf{v}_{2m})\ ]\,.\end{aligned}$$

We now present our description of the sZKAoK proof system based on the same techniques as provided in [24]. Via the Fiat-Shamir heuristic one obtains a signature of knowledge, where the challenge is computed as $\{CH_j\}_{j=1}^c = \mathcal{H}(\mathsf{msg}, \{CMT_j\}_{j=1}^c, \mathsf{pp})$ containing all public parameters pp such as $\mathbf{u}, \mathbf{b}, \hat{\mathbf{A}}, \{\hat{\mathbf{A}}_i\}_{i=0}^{\ell}$.

**Commitments**

- Generate masking terms $\{\mathbf{r}_{\mathbf{e}}^j \hookleftarrow \mathbb{Z}_q^{3n}\}_{j=1}^k$, $\{\mathbf{r}_i^j \hookleftarrow \mathbb{Z}_q^{3n}\}_{j=1}^k$ for $i \in [2m]_{\cup 0}$ and $j \in [k]$ and $\mathbf{r}_{\mathsf{id}^*} \hookleftarrow \mathbb{Z}_q^{2\ell}$
- Sample permutations $\tau \hookleftarrow \mathsf{S}_{2\ell}, \{\psi_j \hookleftarrow \mathsf{S}_{3n}\}_{j=1}^k, \{\pi_j \hookleftarrow \mathsf{S}_{3n}\}_{j=1}^k$, $\{\varphi_j \hookleftarrow \mathsf{S}_{3n}\}_{j=1}^k$

The prover P generates commitments $CMT = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ and sends them to the verifier V. Let $\mathbf{r}^j = [\mathbf{r}_0^j, \ldots, \mathbf{r}_{(2+2\ell)m}^j]$ and denote $\mathbf{D} = [\ \mathsf{rot}(H(m, \mu)) \mid \mathbf{0}\ ] \in \mathbb{Z}_q^{n \times 3n}$.

- $\mathbf{c}_1 = \mathsf{COM}([\ \sum_{i=0}^{(2+2\ell)m} \mathbf{B}_i^* \cdot (\sum_{j=1}^k \beta_j \mathbf{r}_i^j), \mathbf{D} \cdot (\sum_{j=1}^k \beta_j \mathbf{r}_0^j) + [\ \mathbf{I} \mid \mathbf{0}] \cdot (\sum_{j=1}^k \beta_j \mathbf{r}_{\mathbf{e}}^j),\ \tau,\ \{\pi_j\}_{j=1}^k,\ \{\phi_j\}_{j=1}^k,\ \{\varphi\}_{j=1}^k)$
- $\mathbf{c}_2 = \mathsf{COM}(\{[\pi_j(\mathbf{r}_0^j), \ldots, \pi_j(\mathbf{r}_m^j), \psi_j(\mathbf{r}_{m+1}^j), \ldots, \psi_j(\mathbf{r}_{2m}^j), \psi_j(\mathbf{r}_{(\tau(1)+1)m+1}^j), \ldots, \psi_j(\mathbf{r}_{(\tau(1)+2)m}^j), \ldots, \psi_j(\mathbf{r}_{(\tau(2\ell)+1)m+1}^j),$

$\ldots, \psi_j(\mathbf{r}^j_{(\tau(2\ell)+2)m})]\}_{j=1}^k, \{\varphi_j(\mathbf{r}^j_{\mathsf{e}})\}_{j=1}^k, \tau(\mathbf{r}_{\mathsf{id}^*}))$

- $\mathbf{c}_3 = \mathsf{COM}(\{[\pi_j(\mathbf{v}^j_0), \ldots, \pi_j(\mathbf{v}^j_m), \psi_j(\mathbf{v}^j_{m+1}), \ldots, \psi_j(\mathbf{t}^j_{2m}),$
$\psi_j(\mathbf{v}^j_{(\tau(1)+1)m+1}), \ldots, \psi_j(\mathbf{v}^j_{(\tau(1)+2)m}), \ldots, \psi_j(\mathbf{v}^j_{(\tau(2\ell)+1)m+1}),$
$\ldots, \psi_j(\mathbf{v}^j_{(\tau(2\ell)+2)m})]\}_{j=1}^k, \{\varphi_j(\mathbf{v}^j_{\mathsf{e}})\}_{j=1}^k, \tau(\mathbf{v}_{\mathsf{id}^*})\}$.
where $\mathbf{v}^j_i = \mathbf{z}^j_i + \mathbf{r}^j_i$, $\mathbf{v}^j_{\mathsf{e}} = \mathbf{e}^j + \mathbf{r}^j_{\mathsf{e}}$ and $\mathbf{v}_{\mathsf{id}^*} = \mathsf{id}^* + \mathbf{r}_{\mathsf{id}^*}$ .

**Challenge**: The verifier generates a challenge $CH \hookleftarrow \{1, 2, 3\}$ uniformly at random.
**Response**: The response is computed by the prover depending on the outcome of $CH$.

We differentiate 3 cases:

- $\mathbf{CH = 1}$: The response is composed as follows. First, determine $\{\mathbf{b}^j_i = \pi_j(\mathbf{r}^j_i)\}_{j=1}^k$ and $\{\mathbf{w}^j_i = \pi_j(\mathbf{z}^j_i)\}_{j=1}^k$ for $i \in [m]_{\cup 0}$, $\{\mathbf{b}^j_i = \psi_j(\mathbf{r}^j_i)\}_{j=1}^k$ and $\{\mathbf{w}^j_i = \pi_j(\mathbf{z}^j_i)\}_{j=1}^k$ for $m < i \leq 2m$, $\{\mathbf{b}^j_{(l+2)m+i} = \psi_j(\mathbf{r}^j_{(\tau(l))+2)m+i})\}_{j=1}^k$ and $\{\mathbf{w}^j_{(l+2)m+i} = \pi_j(\mathbf{z}^j_{(\tau(l)+2)m+i})\}_{j=1}^k$ for $1 \leq i \leq m$ and $1 \leq l \leq 2\ell$. Furthermore, compute $\mathbf{w}_{\mathsf{id}^*} = \tau(\mathsf{id}^*)$, $\mathbf{b}_{\mathsf{id}^*} = \tau(\mathbf{r}_{\mathsf{id}^*})$ and $\{\mathbf{w}^j_{\mathsf{e}} = \varphi_j(\mathbf{e}^j)\}_{j=1}^k, \{\mathbf{b}^j_{\mathsf{e}} = \varphi_j(\mathbf{r}^j_{\mathsf{e}})\}_{j=1}^k$. Output
$$RSP_1 = \{\{\mathbf{b}^j_0, \mathbf{w}^j_0\}_{j=1}^k, \ldots, \{\mathbf{b}^j_{(2+2\ell)m}, \mathbf{w}^j_{(2+2\ell)m}\}_{j=1}^k, \{\mathbf{w}^j_{\mathsf{e}}, \mathbf{b}^j_{\mathsf{e}}\}_{j=1}^k, \mathbf{w}_{\mathsf{id}^*}, \mathbf{b}_{\mathsf{id}^*}\}.$$
- $\mathbf{CH = 2}$: The prover needs only to output the response
$$RSP_2 = \{\tau, \{\psi_j\}_{j=1}^k, \{\pi_j\}_{j=1}^k, \{\varphi_j\}_{j=1}^k, \{[\mathbf{r}^j_0, \ldots, \mathbf{r}^j_{(2+2\ell)m}]\}_{j=1}^k, \{\mathbf{r}^j_{\mathsf{e}}\}_{j=1}^k, \mathbf{r}_{\mathsf{id}^*}\}.$$
- $\mathbf{CH = 3}$: For $i \in [(2 + 2\ell)m]_{\cup 0}$ compute $\{\mathbf{v}^j_i = \mathbf{z}^j_i + \mathbf{r}^j_i\}_{j=1}^k$. Determine $\mathbf{v}_{\mathsf{id}^*} = \mathsf{id}^* + \mathbf{r}^*_{\mathsf{id}}$ and $\{\mathbf{v}^j_{\mathsf{e}} = \mathbf{e}^j + \mathbf{r}^j_{\mathsf{e}}\}_{j=1}^k$. The response is then given by
$$RSP_3 = \{\tau, \{\psi_j\}_{j=1}^k, \{\pi_j\}_{j=1}^k, \{\varphi_j\}_{j=1}^k, \{\mathbf{v}^j_0\}_{j=1}^k, \ldots, \{\mathbf{v}^j_{(2+2\ell)m}\}_{j=1}^k, \mathbf{v}_{\mathsf{id}^*}\}.$$

**Verification** Via the inputs of the prover, the verifier can always check 2 out of 3 commitments, since responses to all 3 commitments already reveal the witness.

### C. Discussion: DAA and Removing the State

In the previous sections we gave a construction that is secure in a complex security model [13] adopting the UC framework. In fact, we were able to construct a full DAA scheme from lattice assumptions that allows the TPM and host to be in different states each owning its partial private key share, where both interact in order to generate anonymous attestations. A great deal of efforts has been spent in the past years to improve the security model, where most of them failed to offer the appropriate model until recently [13]. The Pre-DAA scheme [5] based on classical assumptions only considers the TPM and host as one party not covering the case where an honest TPM is in a corrupt host (cf. discussions in [13]). In many of the proposed constructions in the literature (and related ones) the TPM owns the whole private key. This was indeed the very first way to realize DAA until the potential advantages of sharing private keys between the TPM and host has been taken into account. Thus, we note that our proposal can easily be modified such that the TPM owns the complete private key rather than just a share. In this case, attestations are generated solely by the TPM.

We further note that our scheme can be made stateless such that the TPM does not need to keep track on already issued tags that are, for instance, required to realize linkability in case the verifier asks for that. For the sake of simplicity, we discuss this aspect here as it represents an independent add-on that can be realized in different ways. One possibility is to let the TPM generate a large enough seed $\mathbf{r}$, which then also belongs to the private key. From this seed it generates the random values for tags that are directed at some verifiers. More specifically, only the error term is sampled using this seed. There must be a one-to-one relationship between the verifier names and the random values. When using a (deterministic) pseudorandom function, for instance, we can deterministically generate random values for any basename as input. These random values can also deterministically be transformed into error vectors. Whenever the same verifier asks for a linkable tag again, the TPM can generate exactly the same tag that the verifier already knows. As a result, the TPM is relieved from storing the tag.


## Acknowledgements

We would like to thank Liqun Chen and Jan Camenisch for their helpful comments and discussions. The first author was supported by the DFG project P1 within the CRC 1119 CROSSING.


## References


[1] Trusted computing group: TPM main specification 1.2. 2004.
[2] Trusted computing group: Trusted platform module library specification, family "2.0". 2014.
[3] M. Ajtai. Generating hard instances of lattice problems (extended abstract). STOC '96, pages 99–108. ACM, 1996.
[4] Rachid El Bansarkhani and Johannes A. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In *Selected Areas in Cryptography - SAC 2013*, pages 48–67, 2013.
[5] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *International Journal of Information Security*, 12(3):219–249, 2013.
[6] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *PKC 2010*, pages 499–517. Springer, 2010.
[7] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. STOC '13, pages 575–584. ACM, 2013.
[8] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 132–145. ACM, 2004.
[9] Ernie Brickell, Liqun Chen, and Jiangtao Li. A new direct anonymous attestation scheme from bilinear maps. In *International Conference on Trusted Computing*, pages 166–178. Springer, 2008.
[10] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, 2009.
[11] Ernie Brickell and Jiangtao Li. Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 21–30. ACM, 2007.
[12] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *International Conference on Trust and Trustworthy Computing*, pages 1–20. Springer, 2016.

[13] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In *IACR International Workshop on Public Key Cryptography*, pages 234–264. Springer, 2016.
[14] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. IEEE, 2001.
[15] Liqun Chen. A daa scheme requiring less tpm resources. In *International Conference on Information Security and Cryptology*, pages 350–365. Springer, 2009.
[16] Liqun Chen, Paul Morrissey, and Nigel P Smart. On proofs of security for daa schemes. In *International Conference on Provable Security*, pages 156–175. Springer, 2008.
[17] Liqun Chen, Paul Morrissey, and Nigel P Smart. Daa: Fixing the pairing based protocols. *IACR Cryptology ePrint Archive*, 2009:198, 2009.
[18] Liqun Chen, Dan Page, and Nigel P Smart. On the design and implementation of an efficient daa scheme. In *International Conference on Smart Card Research and Advanced Applications*, pages 223–237. Springer, 2010.
[19] Marc Fischlin. *Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors*, pages 152–168. Springer, 2005.
[20] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT 2010*, pages 395–412. Springer, 2010.
[21] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. *Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems*. Springer, 2008.
[22] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT 2013*, pages 41–61. Springer, 2013.
[23] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *PKC 2014*, pages 345–361. Springer, 2014.
[24] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC 2015*, pages 427–449. Springer, 2015.
[25] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, pages 1–23. Springer, 2010.
[26] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, pages 700–718. Springer, 2012.
[27] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measure. In *FOCS 2004*, pages 371–381. IEEE, 2004.
[28] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *PKC 2015*, pages 401–426. Springer, 2015.
[29] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 05*, pages 84–93. ACM Press, 2005.
[30] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015*, pages 755–784. Springer, 2015.

## VI. Appendix

## VII. Security Proofs

*Proof (sketch of Theorem 5.1).* We will proceed via a sequence of games in a similar fashion as in [13], [12] for DAA protocols and show therein that there exists no environment $\mathcal{E}$ that can distinguish the real world protocol $\Pi_{daa}$ with an adversary $\mathcal{A}$, from the ideal world $\mathcal{F}_{daa}^{\ell}$ with a simulator $\mathcal{S}$. Starting with the real world protocol, we change the protocol game by game in a computationally indistinguishable way finally resulting in the ideal world protocol. To this end, we introduce an entity $\mathcal{C}$, which first executes $\Pi_{daa}$ for all honest parties. Subsequently and unnoticeable to $\mathcal{A}$, it is split into a simulator $\mathcal{S}_{\mathcal{C}}$ and a functionality $\mathcal{F}_{\mathcal{C}}$, which takes all inputs from and forwards all outputs to the honest parties. The functionality $\mathcal{F}_{\mathcal{C}}$ and the associated simulator $\mathcal{S}_{\mathcal{C}}$ are step by step changed leading to the aimed ideal functionality $\mathcal{F}_{daa}^{\ell}$ and an associated simulator $\mathcal{S}$.

**Game 1**: This is the real world protocol.
**Game 2**: A new entity $\mathcal{C}$ is introduced, which receives all inputs from honest parties and simulates the real world for them, i.e. Game 1 is indistinguishable from Game 2 by construction.
**Game 3**: As explained, $\mathcal{C}$ is split into $\mathcal{F}_{\mathcal{C}}$ and $\mathcal{S}_{\mathcal{C}}$ evolving to become $\mathcal{F}_{daa}^{\ell}$ and $\mathcal{S}$. The functionality $\mathcal{F}_{\mathcal{C}}$ receives and forwards all inputs (authenticated and immediate) to $\mathcal{S}_{\mathcal{C}}$, which simulates the real world protocol for honest parties and sends all outputs to $\mathcal{F}_{\mathcal{C}}$, which forwards them to $\mathcal{E}$. The adversary will notice no difference to Game 2, since beside the structure nothing changed.
**Game 4**: Now, $\mathcal{F}_{\mathcal{C}}$'s behavior is changed such that it also stores the algorithms for $\mathcal{I}$ in the setup interface and further checks for an honest issuer, if sid is built correctly. If not, $\mathcal{F}_{\mathcal{C}}$ will abort. In case $\mathcal{I}$ is corrupt, $\mathcal{S}_{\mathcal{C}}$ extracts the secret key and invokes the setup interface in place of $\mathcal{I}$. Clearly, both cases will not change the view of $\mathcal{E}$, since the input and output behavior is exactly as in Game 3.
**Game 5**: The functionality $\mathcal{F}_{\mathcal{C}}$ also performs verification and linking checks and does not forward the associated queries to $\mathcal{S}_{\mathcal{C}}$. Both algorithms do not require any protocol messages and can be accomplished by anyone, hence the outputs are exactly the same as in the real world. However, $\mathcal{F}_{\mathcal{C}}$ performs private-revocation checks separately leading to the same result as in Game 4.
**Game 6**: In this game, $\mathcal{F}_{\mathcal{C}}$ stores the members that joined and further stores for an honest issuer $\mathcal{I}$ the secret keys gsk of corrupt TPMs, that have been extracted from the proof $\pi$ by $\mathcal{S}_{\mathcal{C}}$ during the simulation of the real world. (Extraction via rewinding requires the inclusion of a fresh nonce to each proof such that the proof is always fresh and extraction is possible. The number of simultaneous sessions for an honest $\mathcal{I}$ is at most logarithmic in the security parameter.) In almost all cases $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{F}_{\mathcal{C}}$ can produce the same outputs as in the real world, since the information suffices to act on behalf of any participant. However, in case $\mathcal{I}$ is honest and all other participants are corrupt, $\mathcal{S}_{\mathcal{C}}$ does not know who triggered join queries and can thus not act with $\mathcal{F}_{\mathcal{C}}$ on behalf of the host. Since $\mathcal{F}_{daa}^{\ell}$ is only concerned with honest hosts in Members, $\mathcal{F}_{\mathcal{C}}$ can safely choose any corrupt host and put it into Members. $\mathcal{F}_{\mathcal{C}}$ does not prohibit a previously allowed query. In the only case, where $\mathsf{tpm}_i$ is already registered and $\mathcal{I}$ is honest, $\mathcal{F}_{\mathcal{C}}$ may abort the protocol. This testing has already been accomplished by $\mathcal{I}$ before continuing with the query JOINPROCEED, hence $\mathcal{F}_{\mathcal{C}}$ will not abort. Since $\mathcal{S}_{\mathcal{C}}$ can simulate the real world protocol in interaction with $\mathcal{F}_{\mathcal{C}}$, Game 5 is indistinguishable from Game 6. Within the next four games $\mathcal{F}_{\mathcal{C}}$ is endowed with the capability to handle signing queries. Previously, it was forwarding all these queries to $\mathcal{S}_{\mathcal{C}}$. We note that each output of $\mathcal{F}_{\mathcal{C}}$ has to be inspected by $\mathcal{S}_{\mathcal{C}}$ such that it can still coordinate/block the output of $\mathcal{F}_{\mathcal{C}}$ in order to match the real world protocol.
**Game 7**: In this game, $\mathcal{F}_{\mathcal{C}}$ creates anonymous signatures for honest platforms using the algorithms defined in the setup interface. We now show gradually that $\mathcal{E}$ can not make any

difference to the previous game.

In Game $7.k.k'$, $\mathcal{F}_\mathcal{C}$ forwards all signing queries with $\mathsf{tpm}_i$ and $i > k$ to $\mathcal{S}_\mathcal{C}$, who creates these signatures as before. $\mathcal{F}_\mathcal{C}$ handles all signing queries with $\mathsf{tpm}_i$ and $i < k$ using the algorithms. The first $k'$ signing inputs with $\mathsf{tmp}_k$ are handled by $\mathcal{F}_\mathcal{C}$. Subsequent inputs are then again forwarded to $\mathcal{S}_\mathcal{C}$. Clearly, we have then Game 7.0.0 = Game 6. One notes that for increasing $k'$, Game $7.k.k'$ will be at some point equal to Game $7.k + 1.0$ as there can only be a polynomial number of signing queries to be processed. With this relationship and for large enough $k$ and $k'$ $\mathcal{F}_\mathcal{C}$ handles all TPMs such that Game 7=Game $7.k.k'$. Therefore, we need only to show that Game $7.k.k' + 1$ is indistinguishable from Game $7.k.k'$. It then directly follows Game 7 is indistinguishable from Game 6.

To this end, we prove that if there exists a distinguisher for Game $7.k.k' + 1$ and Game $7.k.k'$, then we can solve decision (Ring-)LWE. The simulator $\mathcal{S}_\mathcal{C}$ interacting with $\mathcal{F}_\mathcal{C}$ is modified and parametrized with $k, k'$ such that we have Game $7.k.k'$ if it gets a (Ring-)LWE sample, otherwise we have Game $7.k.k' + 1$. Suppose there exists an environment that can distinguish a signature of an honest party using $\mathsf{gsk}$ from a signature using a different $\mathsf{gsk}'$, then we can use the environment as a solver for the decision (Ring-)LWE problem. This is possible, since assuming the hardness of (Ring-)LWE it is even not possible to distinguish many (Ring-)LWE samples generated with the same secret key from uniform random samples. As a result it must follow that (Ring-)LWE samples generated with different keys must be indistinguishable. In fact, this has been shown in Theorem 3.5. Thus, suppose $\mathcal{S}_\mathcal{C}$ is given tuples $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^{k'}, (\mathbf{c}, \mathbf{d})$, where $\mathbf{b}_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i$ for uniform random $\mathbf{a}_i, \mathbf{c} \in \mathcal{R}_q$, and it is challenged to decide, if $(\mathbf{c}, \mathbf{d})$ is from the (Ring-)LWE distribution (for secret $\mathbf{s}$) or uniform random. $\mathcal{S}_\mathcal{C}$ proceeds as follows in order to simulate the TPM without knowing the secret $\mathbf{s}$. Since $\mathcal{S}_\mathcal{C}$ is controlling $\mathcal{F}_{\mathsf{crs}}$, it answers $H$ queries on $\mathsf{bsn}_j$, if not already set, with $H(\mathsf{bsn}_j) = \mathbf{a}_j$ for $j \leq k'$. For $j = k' + 1$, it sets $H(\mathsf{bsn}_{k'}) = \mathbf{c}$, otherwise $H(\mathsf{bsn}_j) = \mathbf{r}_j$ for uniform random $\mathbf{r}_j$ and $j > k' + 1$. Since $\mathcal{S}_\mathcal{C}$ simulates $\mathsf{tpm}_k$, it generates $(\mathbf{r}, \mathbf{u}', \mathsf{id}, \pi_1)$ for uniform random $\mathbf{r}, \mathbf{u}'$ and a simulated proof $\pi_1$ in the join protocol. Signing queries on behalf of $\mathsf{tpm}_i$ for $i < k$ are forwarded by $\mathcal{F}_\mathcal{C}$ to $\mathcal{S}_\mathcal{C}$, which calls the real world protocol. $\mathcal{F}_\mathcal{C}$ signs on behalf of $\mathsf{tpm}_i$ with the algorithms provided by $\mathcal{S}_\mathcal{C}$ for $i > k$, where the $\mathsf{gsk}$s are always freshly sampled for each $\mathsf{bsn}_i$. However, for $\mathsf{tpm}_k$ and $i \leq k'$, the simulator $\mathcal{S}_\mathcal{C}$ sets $\mathsf{nym} = \mathbf{b}_i$. And for $i = k' + 1$ it sets $\mathsf{nym} = \mathbf{d}$. For $i > k' + 1$ it samples fresh $\mathbf{s}_i$ and generates $\mathsf{nym}_i = H(\mathsf{bsn}_i) \cdot \mathbf{s}_i + \mathbf{e}_i$ keeping track of all generated $\mathsf{nym}_i$ (due to stateful tags) such that it can always output the same $\mathsf{nym}_i$ for an associated $\mathsf{bsn}_i$. For each case, $\mathsf{tpm}_k$ can provide a simulated proof. By the $f-\mathsf{IND}$ property of $\mathsf{TAG}_{\mathsf{LIT}}$ any distinguisher between Game $7.k.k'$ and Game $7.k.k' + 1$ can solve decision (Ring-)LWE.

**Game 8**: As opposed to previous games, now $\mathcal{F}_\mathcal{C}$ does not inform $\mathcal{S}_\mathcal{C}$ on $(\mathsf{msg}, \mathsf{bsn})$ used to generate signatures. In case both the Host and the TPM are honest $\mathcal{S}_\mathcal{C}$ is not aware of this pair in order to simulate the real world, instead learns only the leakage $\ell(\mathsf{msg}, \mathsf{bsn})$ over the secure channel. To this end, $\mathcal{S}_\mathcal{C}$ selects any dummy $(\mathsf{msg}', \mathsf{bsn}')$ such that the leakage $\ell(\mathsf{msg}', \mathsf{bsn}') = \ell(\mathsf{msg}, \mathsf{bsn})$ is still the same. Thus, $\mathcal{S}_\mathcal{C}$ can still simulate the real world and $\mathcal{A}$ will observe no difference. As a result, Game 9 is indistinguishable from Game 8.

**Game 9**: Now, platforms, that joined, are allowed by $\mathcal{F}_\mathcal{C}$ to sign, if $\mathcal{I}$ is honest. $\mathcal{E}$ will notice no difference as $\mathcal{S}_\mathcal{C}$ is still the same. For a signing query initiated by a TPM, an honest host will in the real world protocol always check that it joined together with that TPM prior to any continuation, otherwise it aborts. Conversely, an honest TPM only generates signatures in interaction with an host, if it has joined with it. Therefore, no difference to the real world protocol is observed. Thus, Game 10 is indistinguishable from Game 9, since $\mathcal{F}_\mathcal{C}$ does not prevent any signing protocols that are possible in the real world.

**Game 10**: In this game, $\mathcal{F}_\mathcal{C}$ performs honesty-corruption checks prior to storing a secret key $\mathsf{gsk} = \hat{\mathbf{y}}$, i.e. either $\mathsf{CheckGskHonestTPM}(\mathsf{gsk}) = 1$ or $\mathsf{CheckGskCorruptTPM}(\mathsf{gsk}) = 1$ holds. This is true as it is only needed to take valid signatures from $\mathsf{VerResults}$ into account, since $\mathsf{Signed}$ only contains $\perp$ such that $\mathsf{identify}(\perp) = 0$ for an honest TPM and corrupt host or valid signatures of honest hosts and TPMs. Valid signatures satisfy $\mathsf{nym} = H(\mathsf{bsn}) \cdot \mathbf{y}_1 + \mathbf{e}$, where $\|\mathbf{e}\|, \|\mathbf{y}_1\| \leq \beta_1$ and $\|\hat{\mathbf{y}}\| \leq \beta_1$. In fact, there exists only one tuple $(\mathbf{y}_1, \mathbf{e})$ satisfying this relation, as this represents an instance of a (hard) BDD problem or (unique-SVP Problem) for a not too large $\beta_2$. We also refer to Lemma 3.7 for the security of the linkability property. Thus, $\mathsf{CheckGskCorruptTPM}$ will always give the correct output. Due to the large min-entropy of discrete Gaussians the probability of two vectors being equal is negligible. Thus, with overwhelming probability there does not exist a signature already using $\mathsf{gsk}$, which implies that $\mathsf{CheckGskHonestTPM}$ will always give the correct output. From this it follows that Game 9 is indistinguishable from Game 10.

**Game 11**: In this game $\mathcal{F}_\mathcal{C}$ further checks (as done in the signature generation interface of $\mathcal{F}_{daa}^{\ell}$) that honestly generated signatures are valid. Indeed, this is true as $\mathsf{sig}$ always produces signatures passing the verification checks. Furthermore, those signatures have to satisfy $\mathsf{identify}(\mathsf{gsk}, \sigma, \mathsf{msg}, \mathsf{bsn}) = 1$, which is assured by $\mathcal{F}_\mathcal{C}$ during $\mathsf{sig}$ via $\mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{bsn}), \mathbf{y}_1)$. Finally, it makes sure by means of the internal key records $\mathsf{Members}$ and $\mathsf{DomainKeys}$ that honest users are not using the same $\mathsf{gsk}$. To this end, we prove that this does not happen assuming the hardness of (Ring-)LWE. The TPM is simulated using the unknown secret of (Ring-)LWE samples $\{(\mathbf{a}_i, \mathbf{b}_i)\}_{i=1}^{poly(n)}$. $\mathcal{F}_\mathcal{C}$ creates a credential on a uniform random value $\mathbf{r} \in \mathcal{R}_q$ and $\mathbf{b}_1$, sets $\mathsf{nym} = \mathbf{b}_2$ and programs the random oracle $H(\mathsf{bsn}_\mathcal{I}) = \mathbf{a}_1$ and $H(\mathsf{bsn}) = \mathbf{a}_2$ by means of $\mathcal{S}_\mathcal{C}$ for $\mathsf{bsn}$ and finally simulates a matching proof $\pi_2$. If the process is repeated for different $\mathsf{bsn}_i$, it can use the other samples, which are generated from the same secret. If there is a key $\mathsf{gsk} = \hat{\mathbf{y}}$ in $\mathsf{Members}$ and $\mathsf{DomainKeys}$ such that $\|\mathsf{nym} - H(\mathsf{bsn}) \cdot \mathbf{y}_1\| \leq \beta$ we break search (Ring-)LWE. Thus, all checks pass except with negligible probability.

Hence Game 11 is indistinguishable from Game 10. In the following games we successively include further checks from the verification interface of $\mathcal{F}_{daa}^{\ell}$.

**Game 12**: Within the first check multiple gsk values matching one single signature are identified (Check−IX). If exists, the signature is discarded. However, since there exists only one secret and error pair satisfying $\mathsf{nym} = \mathsf{TAG}_{\mathsf{LIT}}(\mathsf{msg}, \mathbf{y}_1)$ (BDD instance for appropriate bounds on the parameter $s_1$ otherwise based on $\mathrm{SIS}_{n,2,\alpha}$ for larger parameters) the check will pass. Thus Game 12=Game 13.

**Game 13**: Now, we further add Check−X to $\mathcal{F}_{\mathcal{C}}$ as it prevents from falsely accepting signatures that were issued by use of join credentials not issued by the honest issuer. In our scheme this can be reduced to existential unforgeability of Boyen signatures or the hardness to solve (Ring-)ISIS. The entity $\mathcal{C}$ registers the Boyen public key together with its associated simulated proof. In case $\mathcal{I}$ has to generate join credentials, it sends the extracted $\mathsf{gsk} = (\hat{\mathbf{y}}, \mathsf{id})$ to the signing oracle, which outputs $\hat{\mathbf{x}}$. $\mathcal{I}$ computes $\mathbf{u}' = \hat{\mathbf{A}}_{\mathsf{id}} \cdot \hat{\mathbf{x}} \bmod q$. In order to generate signatures on behalf of honest platforms, for which $\mathcal{F}_{\mathcal{C}}$ stores gsk in Members or DomainKeys, $\mathcal{F}_{\mathcal{C}}$ uses the signing oracle. For a valid signature $\sigma = \pi_2, \pi_3, \mathbf{u}', \mathsf{nym}, \mathsf{bsn}$ the verifier checks, if there exists an entry gsk in Members or DomainKeys such that $\|\mathsf{nym} - H(\mathsf{bsn}) \cdot \mathbf{y}_1\| \leq \beta$. If not, it extracts by the special soundness of the proofs $\hat{\mathbf{z}} + (\mathbf{0}, \hat{\mathbf{t}}), \mathsf{id}^{(1)}$ and $(\mathbf{0}, \hat{\mathbf{x}} - \hat{\mathbf{t}}), \mathsf{id}^{(2)}$ such that we obtain a Boyen signature, if $\mathsf{id}^{(1)} = \mathsf{id}^{(2)}$, otherwise he must have solved (Ring-)ISIS such that $[\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}^{(1)}}] \cdot (\hat{\mathbf{z}} + (\mathbf{0}, \hat{\mathbf{t}})) + \hat{\mathbf{A}}_{\mathsf{id}^{(2)}} \cdot (\hat{\mathbf{x}} - \hat{\mathbf{t}}) \equiv \mathbf{u} \bmod q$, since $\mathbf{z}_1 \neq \mathbf{y}_1$ and with overwhelming probability there exist no two TPMs with the same syndromes $\tilde{\mathbf{u}} = [\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}}] \cdot \hat{\mathbf{z}}$, where one of them is honest. Furthermore, changing $\mathbf{y}_1$ to $\mathbf{y}_1 + \Delta$ of a corrupt identity $\mathsf{id}^{(i)}$ requires to solve (Ring)ISIS w.r.t. an identity $\mathsf{id}^{(j)}$ in order to satisfy the equation, i.e. finding an admissible $\hat{\mathbf{x}}'$ such that $\mathbf{b} \cdot \Delta + [\mathbf{b}, \hat{\mathbf{A}}_{\mathsf{id}^{(i)}}] \cdot \hat{\mathbf{y}} + \hat{\mathbf{A}}_{\mathsf{id}^{(j)}} \cdot \hat{\mathbf{x}}' \equiv \mathbf{u} \bmod q$. We refer to Remark III-B for further details. Since Boyen signatures are unforgeable and the computational problem (Ring-)ISIS is hard to solve, Game 13=Game14.

**Game 14,15**: First, Check−XI is added to $\mathcal{F}_{\mathcal{C}}$. In fact, the goal of $\mathcal{F}_{\mathcal{C}}$ is to prevent any signatures that are signed using the key and credentials of an honest TPM, but the TPM never signed. We show that providing such a signature can be reduced to solving search (Ring-)LWE. To this end, the TPM is simulated by use of the unknown secret of a (Ring-)LWE instance. Then, if a valid signature is given on a message, that the TPM never signed, the proof could not have been simulated. It extracts $\mathbf{y}_1$ and thus breaks search (Ring-)LWE. Finally, we add Check−XII to $\mathcal{F}_{\mathcal{C}}$, which prohibits the revocation of an honest TPM. This task can also be reduced to solving search (Ring-)LWE as the honest TPM is simulated by means of the (Ring-)LWE problem instance (and a uniform random element). If a proper key in the revocation list is found, it must be the secret key of the target instance.

**Game 16**: All remaining checks of $\mathcal{F}_{daa}^{\ell}$ are included, which are related to link queries. $\mathcal{F}_{\mathcal{C}}$'s outcome on a gsk that matches one signature and not the other is 0 indicating that the signatures are not linked. If both signatures match to one gsk, then $\mathcal{F}_{\mathcal{C}}$ outputs 1 indicating that the signatures are linked. These outputs match the output of the deterministic algorithm link. This stems from the fact that if a gsk matches one signature and not the other, then $\mathsf{nym} \neq \mathsf{nym}'$ and link outputs 0 due to the soundness of the proofs. And if both signatures match to one gsk, then $\|\mathsf{nym} - \mathsf{nym}'\| \leq \sqrt{2}\beta_1$ by the soundness of the proof and link outputs 1. Thus, Game 16 is indistinguishable from Game 15.

$\mathcal{F}_{\mathcal{C}}$ now includes all functionalities of $\mathcal{F}_{daa}^{\ell}$. This concludes our proof. □