A Framework for Efficient Lattice-Based DAA

Liqun Chen
University of Surrey
liqun.chen@surrey.ac.uk

Nada El Kassem
University of Surrey
n.elkassem@surrey.ac.uk

Anja Lehmann
IBM Research – Zurich
anj@zurich.ibm.com

Vadim Lyubashevsky
IBM Research – Zurich
vad@zurich.ibm.com

ABSTRACT

Currently standardized Direct Anonymous Attestation (DAA) schemes have their security based on the factoring and the discrete logarithm problems, and are therefore insecure against quantum attackers. This paper presents a quantum-safe lattice-based Direct Anonymous Attestation protocol that can be suitable for inclusion in a future quantum-resistant TPM. The security of our proposed scheme is proved in the Universal Composability (UC) model under the assumed hardness of the Ring-SIS, Ring-LWE, and NTRU problems. The signature size of our proposed DAA scheme is around 2MB, which is (at least) two orders of magnitude smaller compared to existing post-quantum DAA schemes.

ACM Reference Format:
Liqun Chen, Nada El Kassem, Anja Lehmann, and Vadim Lyubashevsky. 2019. A Framework for Efficient Lattice-Based DAA. In *1st Workshop on Cyber-Security Arms Race (CYSARM'19), November 15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3338511.3357349

1 INTRODUCTION

Direct Anonymous Attestation (DAA) is a cryptographic protocol that allows a Trusted Platform Module (TPM) to serve as a trust anchor for a host platform it is embedded in. To do so, the TPM chip creates attestations about the state of the host system, e.g., certifying the boot sequence the host is running on. These attestations convince a remote verifier that the platform it is communicating with is running on top of trusted hardware and using the correct software. A main design goal of DAA is that attestations are made in a *privacy-preserving* manner. That is, the verifier can check that attestations originate from a certified hardware token, but it does not learn anything about the identity of the particular TPM. Another important feature of DAA is that it supports *user-controlled linkability* which is steered by a basename *bsn*. If a platform uses a fresh or empty basename, the resulting attestations cannot be linked whereas repeated use of the same basename makes the transactions linkable.

CYSARM'19, November 15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6840-7/19/11…$15.00
https://doi.org/10.1145/3338511.3357349

Overall, DAA can be seen as a special variant of group signatures with a central issuer controlling membership to the group of certified TPMs, and TPMs being able to sign anonymously on behalf of the group. Instead of the opening capabilities provided in group signatures, DAA controls privacy through the use of basenames and user-controlled linkability.

DAA has been developed for the Trusted Computing Group (TCG), which is the industry group that designs the TPM. The first RSA-based DAA protocol got standardized in the TPM 1.2 specification [Tru04] in 2004 [BCC04], whereas the newer TPM 2.0 standard [Tru14] published in 2014 supports a suite of elliptic-curve based DAA protocols [BCL08, BCL09, CPS10] that are specified in the complementing ISO standard ISO/IEC 20008-2 [Int13, Int15]. As reported by TCG, more than a billion devices include TPM technology; in particular almost all enterprise PCs, many servers and embedded systems rely on such trusted hardware anchors. Since the first proposal of DAA, many extensions and works to improve security and efficiency have been proposed [BFG+13, CU15, CDL16a, XYZF14, CDL16b, CDL17, CCD+17]. A variant of DAA called Enhanced Privacy ID (EPID) is used in Intel SGX [BL07], the most advanced development in the area of trusted computing.

Recently, the practical interest in DAA has revived, as providing authenticity of attestations while preserving the privacy of senders is enjoying increased attention and awareness. Anonymity of attestations is particularly important in automotive applications such vehicle-to-vehicle communication, wherein tracking of drivers should be prevented but authenticity of the communication must be guaranteed too [WCG+17]. A DAA protocol has also been integrated into the Fast IDentity Online (FIDO) authentication framework [CDE+]. In this application, the TPM creates a new authentication key, and outputs a DAA signature in order to certify that the key is properly stored in the TPM. Another DAA-based application is a privacy-enhancing cloud service architecture to protect user's data, using DAA to let users control the extent of data sharing among their service accounts [GJL11].

The existing DAA schemes that are currently supported by the TPMs are based on either the factorization problem in the RSA setting or the discrete logarithm problem in the Elliptic Curve (EC) setting. Since the factorization problem and discrete logarithm problem are known to be vulnerable to quantum computer attacks, all standardized DAA protocols are not post-quantum secure, i.e. an adversary with a powerful quantum computer could break the TPM's security.

Thus, there is a need to update the cryptographic primitives of current privacy-preserving schemes to be quantum resistant. Many proposed post-quantum cryptographic primitives are build

on the top of code-, hash-, lattice- and multivariate-based problems, and could possibly be used as the basis for the development of post-quantum DAA protocols. Among these, lattice-based cryptography seems to be the most flexible one, supporting several variants of anonymous signature schemes, such as lattice-based group signatures [LLM+16, LNW15, dPLS18], and ring signatures [GCH+19, BS15]. Recently, El Kassem et al. [KCB+19] and El Bansarkhani and El Kaafarani [BK17] proposed two post-quantum DAA scheme from lattice assumptions, however both schemes require massive storage and computation resources, which makes them not suitable for inclusion in the future quantum-resistant TPM.

1.1 Our Results

Our DAA is related to the recently-proposed group signature scheme [dPLS18], which at the time of this writing is the most efficient quantum-safe scheme for large group sizes. The high-level idea of the group signature scheme is for the issuer to create user secret keys by outputting signatures (where messages are the user's identity) of the ABB lattice-based signature scheme [ABB10]. More precisely, the secret key of a group member with identity $i \in \mathbb{Z}_q^*$ is an ABB signature $\mathbf{s}$ of the message i. In other words, $\mathbf{s}$ was a small-norm polynomial satisfying

$$[\mathbf{A} \mid \mathbf{B} + i\mathbf{G}]\mathbf{s} = \mathbf{u}, \tag{1}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{G}$, and $\mathbf{u}$ are public parameters over some polynomial ring. To construct a signature of a message μ, the group member encrypts (a part of) $\mathbf{s}$ and gives a non-interactive ZKPoK (using μ in the hash of the challenge) to prove knowledge of the $i \in \mathbb{Z}_q^*$ and $\mathbf{s}$ satisfying (1)[1] as well as the fact that (a part of) $\mathbf{s}$ was encrypted.

DAA schemes differ from basic group signatures in three ways:

(1) extra privacy properties are required for users in case of a malicious issuer,
(2) the user in a DAA scheme is split into two parts - the TPM and the host - and they do signing in a way that doesn't reveal the TPM's secret (even to the host),
(3) there is no opener, but there is instead a linking procedure that should allow anyone to link two signatures for a common *basename*

Notice that in the above group signature description, the issuer needs to know the i in order to produce the group member's secret key and so there is nothing preventing a malicious issuer from impersonating a group member. This would not satisfy the security definitions of an analogous DAA scheme. To remedy this, instead of directly using the selectively-secure ABB scheme, we use the idea in the Ducas-Micciancio scheme [DM14], which modifies the ABB signature to include a tag τ when signing a low-norm message $\mathbf{m}$. We further modify their scheme to a selectively-secure one (with a small tag domain to still ensure security) which is more efficient. A signature $\mathbf{s}$ of a message $\mathbf{m}$ under a tag τ satisfies

$$[\mathbf{A} \mid \mathbf{B} + \tau\mathbf{G}]\mathbf{s} = \mathbf{u} + \mathbf{a} \cdot \mathbf{m}, \tag{2}$$

where $\mathbf{a}$ is now an additional public parameter and τ is chosen fresh for every signature that is created.[2] The way in which we utilize this construction by observing that if the group member sends $\mathbf{v} = \mathbf{am}$, along with a proof of knowledge of $\mathbf{m}$, then the authority can create a signature on $\mathbf{m}$ without its knowledge. In this way, the group signature scheme of [dPLS18] can be modified to satisfy the stronger definition of *dynamic* group signatures in which a malicious issuer cannot impersonate a group member.

The above-described dynamic group signature issuance procedure can now be easily converted to an issuance procedure of a DAA scheme. In particular, $\mathbf{m}$ (or at least a part of it) will be the TPM secret key, while $\mathbf{s}$ is the secret membership credential of the host. This information, together with τ, is then used by the TPM and the host to create a ZKPoK, much in the same way as the ZKPoK of (1) was generated in [dPLS18], of (2) to sign messages.

The other difference between group signatures and DAA schemes is that DAA schemes have a "linking" procedure instead of an opening. So instead of encrypting the TPM's secret, we will instead create a Ring-LWE instance $(b, nym = bm_1 + e)$ where b is the basename, m_1 is part of the TPM's secret, and e is an error term that's an output of a PRF evaluated at b and m_1.[3]. To allow efficient linking (i.e. there should be a public procedure that allows to determine whether the same TPM signed under the same basename) we need to take care about how the TPM proves knowledge of m_1, e satisfying $nym = bm_1 + e$. If we use the most efficient proof that also proves knowledge of a low-norm $\bar{c}, \bar{m}_1, \bar{e}$ such that $\bar{c} \cdot nym = b\bar{m}_1 + \bar{e}$, then linking would require guessing the $\bar{c}$ (which could come from an exponentially-large space). Instead, we use the slightly less-efficient proof from [BCK+14] which proves the knowledge of $\bar{m}_1, \bar{e}$ satisfying $2 \cdot nym = b\bar{m}_1 + \bar{e}$. Putting this proof together with the proof of (2) (and using the message μ in the hash of the challenge), as well as making several small adjustments to allow the UC proof to go through, completes the signature.

1.1.1 *Comparison to Existing Lattice-Based DAA Schemes.*

In our proposed scheme, the TPM's secret key consists of two short polynomials in the ring $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. In the LDAA scheme [KCB+19], the TPM's secret key consists of $m = 24$ polynomials, and there are $2m + 1 = 49$ polynomials in [BK17]. The degree of our ring should be set to 4096 (as in [dPLS18]), whereas it's possible that the degree of the ring in the other schemes could be 2048 or 1024. Still, our scheme should be faster in terms of the TPM's computation costs in the join and sign interfaces, and have smaller TPM keys and signature sizes.

In our proposed scheme, the signature consists of around 45 polynomials in R_q. Using the same values of $q \approx 2^{70}$ and $d = 4096$ as in [dPLS18] and also account for the fact that most polynomials in R_q have coefficients smaller than q, a rough estimate for the size of the signatures is 2MB.

The LDAA signature includes c responses to the Fiat-Shamir challenges, where each response is comprised of approximately

[1] Actually, one proves a "relaxed" version of (1) involving a lot of extra small-norm polynomial multiplicands due to the fact that the most efficient zero-knowledge proofs for commitments (c.f. [BDL+18]) only prove knowledge of approximate solutions.

[2] The scheme of [dPLS18] is only proved non-adaptive security of their signature schemes where the adversary needs to choose the messages he wants to see the signatures of before seeing the signatures. It is converted to a standard signature scheme using a chameleon hash. We show in our proof how to go around the requirement of the chameleon hash.

[3] The reason for using a PRF to generate the error is that it would be insecure to output $bm_1 + e$ and $bm_1 + e'$ for $e \neq e'$

$km(2\ell+2)$ polynomials provided by the host and $k(m'+1)$ provided by the TPM. In [BK17], the size of the response for each round is bounded by $km(2\ell+2)$ polynomials for both the host and the TPM (where $\ell = 32, m' = 24, k = 8, c = 8$).

Even if the degree of the rings in [KCB+19] and [BK17] are 4× smaller (i.e. the ring dimension is 1024) than in our scheme and the bit-length of the polynomial coefficients are also 4× smaller (the modulus q is less than 2^{18}), the signature produced by our scheme is still more than two order of magnitudes shorter. We should point out that this is still around 5 - 6 orders of magnitude longer than discrete logarithm based DAA schemes (e.g. [CDL16b, CDL17]).

2 PRELIMINARIES

2.1 Notation

$x \leftarrow S$ means that x is a uniformly random sample drawn from a set S. If D is a distribution, then $x \leftarrow D$ means that x is drawn according to the distribution D. We define the ring of polynomials $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$, where d is the dimension of R_q which is a power of 2. We write $c = c_0 + c_1 x + \cdots + c_{d-1}x^{d-1}$ to represent a polynomial in R_q with integer coefficients, $\|c\|_\infty$ denotes the infinity norm of polynomial c, with $\|c\|_\infty = \max_{0\le j\le n} |c_j|$. We will always assume that $q \equiv 3 \pmod 8$, which implies [LN17, Lemma 2.2] that all (non-zero) elements in R_q whose infinity norm is less than $\sqrt{q/2}$ are invertible. Additionally, with probability almost 1, a random element in R_q is also invertible.

$\mathbf{a} = (a_1, \ldots, a_k)$ represents a vector of polynomials in R_q^k, for some positive integer k and polynomials $a_1, \ldots, a_k$ in R_q. $\mathbf{A} \in \mathcal{R}_q^{k\times\ell}$ is a matrix whose entities are polynomials $a_{ij} \in R_q$ for $1 \le i \le k$ and $1 \le j \le \ell$. $\|\mathbf{A}\|_\infty$ is the infinity norm of the matrix of polynomials $\mathbf{A}$ defined by $\|\mathbf{A}\|_\infty = \max_i \|a_{ij}\|_\infty$.

2.2 Lattice Problems and Sampling

For $x, c \in \mathbb{R}^d$ and $\xi \in \mathbb{R}^+$, we define the Gaussian function $\rho_{c,\xi}(x) = \exp\left(\frac{-\|x-c\|^2}{2\xi^2}\right)$, and for a lattice $\mathcal{L}$, we define the distribution $D_{\mathcal{L},c,\xi}(x)$ to be 0 whenever $x \notin \mathcal{L}$ and

$$D_{\mathcal{L},c,\xi}(x) = \frac{\rho_{c,\xi}(x)}{\sum_{v\in\mathcal{L}} \rho_{c,\xi}(v)} \tag{3}$$

when $x \in \mathcal{L}$. When we omit the $\mathcal{L}$ from the above equation, it is assumed that the lattice is $\mathbb{Z}^d$ (where d is evident from context). Omitting the c implies that $c = 0$.

As an additive group, the polynomial ring $R = \mathbb{Z}[X]/(X^d + 1)$ has an obvious mapping to $\mathbb{Z}^d$ and so we can write $v \leftarrow D_\xi$ to signify sampling a random centered element from R.

For a polynomial vector $\mathbf{a} = (a_1, \ldots, a_k) \in R_q^k$ and $t \in R_q$, we can define a k-dimensional shifted lattice[4]

$$\mathcal{L}^\perp_{\mathbf{a},t} = \{\mathbf{s} \in R^k \ : \ a_1 s_1 + \ldots + a_k s_k = t \bmod q\}$$

and we define the distribution $D^\perp_{\mathbf{a},t,\xi}(x)$ to be 0 whenever $x \notin \mathcal{L}^\perp_{\mathbf{a},t}$ and

$$D^\perp_{\mathbf{a},t,\xi}(x) = \frac{\rho_\xi(x)}{\sum_{v\in\mathcal{L}^\perp_{\mathbf{a},t}} \rho_\xi(v)} \tag{4}$$

The Ring-SIS problem [LM06] is defined as finding a short linear combination $s_1, \ldots, s_m \in R_q$ satisfying $a_1 s_1 + \ldots + a_m s_m = 0$ for a given set of randomly-chosen $a_i \in R_q$. The decisional Ring-LWE problem [LPR10] is, for random $a_i, s \leftarrow R_q$ and random small-coefficient polynomials $e_i \in R_q$, to distinguish tuples $(a_i, a_i s + e_i)$ from uniformly random tuples in R_q. Note that because random elements in R_q are invertible with probability close to 1, the triples $(a_i, b_i, a_i s + b_i e_i)$ are also indistinguishable from uniform triples based on Ring-LWE.

In general, given a random $\mathbf{a}, t$, it is hard (as hard as the Ring-SIS problem) to sample according to $D^\perp_{\mathbf{a},t,\xi}$ for small ξ. One can do such sampling, however, when given a special trapdoor basis for the lattice $\mathcal{L}^\perp_{\mathbf{a},0}$. The smaller the vectors in the trapdoor, the smaller the ξ can be in the distribution.

2.3 Lattice Trapdoor Sampling

The trap-door sampler that will be used in our scheme defines $\mathbf{a}$ as a 4-element vector $(\mathbf{a}', \mathbf{a}'\mathbf{R} + (\tau, \tau\lceil\sqrt{q}\rfloor))$, where $\mathbf{a}' = (a, 1)$ for a uniformly-random $a \in R_q$, $\mathbf{R} \in R_q^{2\times 2}$ consists of polynomials with random coefficients in $\{-1, 0, 1\}$ and τ is some non-zero element in $\mathbb{Z}_q$. It was shown in [MP12, Lemma 5.3] that one can sample, again using [GPV08, DP16], elements from $D^\perp_{\mathbf{a},t,\xi}$, for any $t \in R_q$, for $\xi \approx 2(s_1(\mathbf{R}) + 1)\sqrt{q+1}$, where $s_1(\mathbf{R}) = \max_{\mathbf{v}\neq 0} \frac{\|\mathbf{R}\mathbf{v}\|}{\|\mathbf{v}\|}$. For the $\mathbf{R}$ generated as above, $s_1(\mathbf{R})$ is concentrated around $3\sqrt{d}$, and so one can sample from $D^\perp_{\mathbf{a},t,\xi}$ with $\xi \ge 6\sqrt{dq}$. We will refer to this algorithm as the MP-Sampler.

2.4 Zero Knowledge Proofs

Given a matrix $\mathbf{A} \in R_q^{n\times m}$, and a vector $\mathbf{t} \in R_q^n$ satisfying

$$\mathbf{A}\mathbf{r} = \mathbf{t}, \tag{5}$$

for some vector $\mathbf{r}$ with a bounded norm, there are several protocols for proving knowledge of this short $\mathbf{r}$ satisfying a possibly "relaxed" version of (5). The most expensive, in terms of proof size, is a proof that proves exactly the knowledge of (5) for an ℓ_∞ bound on $\mathbf{r}$. This is the adaptation of Stern's proof [Ste93] working over larger rings [KTX08, LNSW13], and the proof sizes are on the order of megabytes. There have been some recent improvements [YAZ+19, BLS19, Beu19, BN19], but the proofs are still on the order of several hundred kilobytes for even relatively short vectors. In our protocol, we will use these proofs in places where a large proof size is not important because the procedure is only done once – i.e. when the issuer registers his public key to the CA and when TPMs perform the join.

A more efficient (in terms of proof size) proof system can prove the knowledge of a slightly larger vector $\bar{\mathbf{r}}$ satisfying $\mathbf{A}\bar{\mathbf{r}} = 2\mathbf{t}$. This scheme from [BCK+14] is presented in Figure 2. The most efficient proof system is based on the digital signature from [Lyu12] and

[4] A *shifted lattice* is a set that is a lattice shifted by some vector v. Note that a shifted lattice does not have the property that the sum of any two vectors is in the shifted lattice.

Prover		Verifier
$\mathbf{A} \in R_q^{n\times m}, \mathbf{r}, \mathbf{t} = \mathbf{Ar}, \xi > \max_{c\in C} 11 \cdot \|c\mathbf{r}\|$		$\mathbf{A}, \mathbf{t}, \beta_z = \xi \cdot \sqrt{2md}$
$\mathbf{y} \leftarrow D_\xi, \mathbf{w} := \mathbf{Ay}$	$\xrightarrow{\mathbf{w}}$	
	$\xleftarrow{c}$	$c \leftarrow C$
$\mathbf{z} := c\mathbf{r} + \mathbf{y}$ if $\mathtt{rej}(\mathbf{z}, c\mathbf{r}, \xi) = 1$, abort	$\xrightarrow{\mathbf{z}}$	Accept iff: $\|\mathbf{z}\| \le \beta_z$, $\mathbf{Az} = c\mathbf{t} + \mathbf{w}$

Figure 1: Proof of Knowledge for $\bar{\mathbf{r}}, \bar{c}$, with $\|\bar{\mathbf{r}}\| \le 2\beta_z$ and $\bar{c} \in \bar{C}$, satisfying $\mathbf{A}\bar{\mathbf{r}} = \bar{c}\mathbf{t}$. $C = \{c \in R_q | \, \|c\|_1 = \kappa$ such that $\log\binom{d}{\kappa} > 256 - \kappa, \|c\|_\infty = 1\}$ and $\bar{C} = C - C$ excluding 0.

proves knowledge of a vector $\bar{\mathbf{r}}$ and a small polynomial c satisfying $\mathbf{A}\bar{\mathbf{r}} = c\mathbf{t}$. This scheme is presented in Figure 1.

All our protocols use a rejection sampling subroutine from [Lyu12] to produce outputs that are distributed as gaussians with distributions independent of the secret key (and the challenge). This is crucial for showing that the protocols are zero-knowledge.

$\mathtt{rej}(\mathbf{z}, \mathbf{b}, \xi)$: $u \leftarrow [0,1)$; if $u > 1/3 \exp\left(\frac{-2\langle \mathbf{z}, \mathbf{b}\rangle + \|\mathbf{b}\|^2}{2\xi^2}\right)$ return 0, else return 1

Lemma 2.1 ([Lyu12]). *Let V be a subset of $\mathbb{R}^n$ with elements of norm less than T, let h be a distribution of V. $\mathbf{b} \in \mathbb{R}^n$. Consider a procedure that samples a $\mathbf{y} \leftarrow D_\xi^n$ and then returns the output of $\mathtt{rej}(\mathbf{z} := \mathbf{y} + \mathbf{b}, \mathbf{b}, \xi)$ where $\xi \ge 11\|\mathbf{b}\|$. The probability that this procedure outputs 1 is $\approx 1/3$. The distribution of $\mathbf{z}$, conditioned on the output being 1, is within statistical distance 2^{-100} of D_ξ^n.*

2.5 Lattice-Based Commitments

In this paper we use the version of the commitment scheme from [BDL+18] that commits to k ring elements. Define the public parameter $\mathbf{C}$ as

$$\mathbf{C} = \begin{bmatrix} a_1 & a_2 & \dots & a_k & a_{k+1} & 1 \\ b_1 & 0 & \dots & 0 & b'_1 & 0 \\ b_2 & 0 & \dots & b'_2 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_k & b'_k & \dots & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \tag{6}$$

where $\mathbf{A}$ is the top row and $\mathbf{B}$ is the rest of the matrix. The commitment of k elements in R_q, $\mathbf{m} = \begin{bmatrix} m_1 \\ \dots \\ m_k \end{bmatrix}$, consists of creating an $\mathbf{r}$ with small (e.g. $-1, 0, 1$)coefficients and outputting

$$\mathbf{Cr} + \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} t_A \\ \mathbf{t}_B \end{bmatrix} = \mathbf{t} \tag{7}$$

From the above definitions, observe that we have $t_A = \mathbf{Ar}$ and $\mathbf{t}_B = \mathbf{Br} + \mathbf{m}$.

Given a public $\mathbf{C}$ and a commitment $\mathbf{t}$ to messages $m_1, \dots, m_k \in R_q$, there is a zero-knowledge proof of knowledge of an $\bar{\mathbf{m}} \in R_q^k$, $\bar{\mathbf{r}}$ with $\|\bar{\mathbf{r}}\| \le 2\beta_z$ (as defined in Figure 3) and $\bar{c} \in \bar{C}$ satisfying

$$\mathbf{C}\bar{\mathbf{r}} + \begin{bmatrix} 0 \\ \bar{c}\bar{\mathbf{m}} \end{bmatrix} = \bar{c}\mathbf{t} \text{ and } \mathbf{L}\bar{\mathbf{m}} = \mathbf{u} \tag{8}$$

Obtaining the above proof is a slight generalization of the proof system from [BDL+18]. What we will need in our proof, however, is additionally to show that $m_i \in \mathbb{Z}_q$. That is, m_i are polynomials in R_q that are only non-zero in their constant coefficient. Proving this additional restriction on $\bar{\mathbf{m}}$ involves using the "automorphism stability" modification from [dPLS18]. The latter work uses the fact that an element $m \in R_q$ is in $\mathbb{Z}_q$ if and only if it satisfies $m = \sigma_{-1}(m) = \sigma_5(m)$, where $\sigma_j(m(X)) = m(X^j)$ [dPLS18]. The proof modification therefore also needs to prove that for all $\bar{m}_i$ composing $\bar{\mathbf{m}}$, $\bar{m}_i = \sigma_{-1}(\bar{m}_i) = \sigma_5(\bar{m}_i)$. We present this proof in Figure 3.

Lemma 2.2. *The protocol in Figure 3 is a proof of knowledge of* (8).

Sketch. To prove zero-knowledge in the case that $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2$ are sent, we observe that due to the rejection sampling procedure the distribution of $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2$ is exactly D_ξ. Therefore one can simulate the view of the verifier by generating $\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2 \leftarrow D_\xi$, $c \leftarrow C$ and setting $w, w_1, w_2, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_L$ according to the verification equations.

To show that the protocol is a proof of knowledge, note that the usual rewinding of the prover after the first step allows us to extract $\bar{\mathbf{z}}, \bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \bar{c}$ satisfying

$$\mathbf{A}\bar{\mathbf{z}} = \bar{c}t_A \tag{9}$$

$$\sigma_{-1}^{-1}(\mathbf{A})\bar{\mathbf{z}}_1 = \bar{c}\sigma_{-1}^{-1}(t_A) \tag{10}$$

$$\sigma_5^{-1}(\mathbf{A})\bar{\mathbf{z}}_2 = \bar{c}\sigma_5^{-1}(t_A) \tag{11}$$

$$\mathbf{B}\bar{\mathbf{z}} - \sigma_{-1}^{-1}(\mathbf{B})\bar{\mathbf{z}}_1 = \bar{c}(\mathbf{t}_B - \sigma_{-1}^{-1}(\mathbf{t}_B)) \tag{12}$$

$$\mathbf{B}\bar{\mathbf{z}} - \sigma_5^{-1}(\mathbf{B})\bar{\mathbf{z}}_2 = \bar{c}(\mathbf{t}_B - \sigma_5^{-1}(\mathbf{t}_B)) \tag{13}$$

$$\mathbf{LB}\bar{\mathbf{z}} = \bar{c}(\mathbf{L}\mathbf{t}_B - \mathbf{u}) \tag{14}$$

We would like to now prove that (9) and (11), along with the SIS assumption, imply that

$$\sigma_5(\bar{c})\bar{\mathbf{z}} = \bar{c}\sigma_5(\bar{\mathbf{z}}_2). \tag{15}$$

To show the above, multiply (9) by $\sigma_5(\bar{c})$, multiply $\sigma_5((11))$ by $\bar{c}$ and subtract to obtain

$$\mathbf{A}(\sigma_5(\bar{c})\mathbf{z} - \bar{c}\sigma_5(\bar{\mathbf{z}}_2)) = 0. \tag{16}$$

Since we assumed that the SIS problem is hard, then the above can only be true if (15) holds.

Because $\bar{c}$ is invertible, it is possible to define $\bar{\mathbf{m}}$ such that

$$\bar{c}\bar{\mathbf{m}} = \bar{c}\mathbf{t}_B - \mathbf{B}\bar{\mathbf{z}} \tag{17}$$

Using (13), we also obtain

$$\bar{c}\bar{\mathbf{m}} = \bar{c}\sigma_5^{-1}(\mathbf{t}_B) - \sigma_5^{-1}(\mathbf{B})\bar{\mathbf{z}}_2. \tag{18}$$

Applying the automorphism σ_5 to the above equation and multiplying by $\bar{c}$, and then applying (15), we get

$$\bar{c}\sigma_5(\bar{c})\sigma_5(\bar{\mathbf{m}}) = \bar{c}\sigma_5(\bar{c})\mathbf{t}_B - \bar{c}\mathbf{B}\sigma_5(\bar{\mathbf{z}}_2) = \bar{c}\sigma_5(\bar{c})\mathbf{t}_B - \sigma_5(\bar{c})\mathbf{B}\bar{\mathbf{z}} \tag{19}$$

Subtracting the above from $\sigma_5(\bar{c})$ times (17), we obtain that

$$\bar{c}\sigma_5(\bar{c})\sigma_5(\bar{\mathbf{m}}) = \bar{c}\sigma_5(\bar{c})\bar{\mathbf{m}},$$

and since both $\bar{c}$ and $\sigma_5(\bar{c})$ are invertible, this implies that $\sigma_5(\bar{\mathbf{m}}) = \bar{\mathbf{m}}$. Exactly the same proof yields that $\sigma_{-1}(\bar{\mathbf{m}}) = \bar{\mathbf{m}}$ and so $\bar{\mathbf{m}} \in \mathbb{Z}_q^k$.

Prover		**Verifier**
$\mathbf{A} \in R_q^{n\times m}, \mathbf{r}, \mathbf{t} = \mathbf{Ar}, \xi > 11 \cdot \sqrt{\ell} \cdot \|\mathbf{r}\|$		$\mathbf{A}, \mathbf{t}, \beta_z = \xi \cdot \sqrt{2md}$
for all $1 \le i \le \ell$: $\mathbf{y}_i \leftarrow D_\xi$, $\mathbf{w}_i := \mathbf{Ay}_i$	$\xrightarrow{\mathbf{w}_1,\ldots,\mathbf{w}_\ell}$	
	$\xleftarrow{c_1,\ldots,c_\ell}$	$c_1, \ldots, c_\ell \leftarrow C_x$
for all $1 \le i \le \ell$: $\mathbf{z}_i := c_i\mathbf{r} + \mathbf{y}_i$		
if $\mathtt{rej}([\mathbf{z}_1 \mid \ldots \mid \mathbf{z}_\ell], [c_1\mathbf{r} \mid \ldots \mid c_\ell\mathbf{r}], \xi) = 1$, abort	$\xrightarrow{\mathbf{z}_1,\ldots,\mathbf{z}_\ell}$	
		Accept iff for all $1 \le i \le \ell$: $\|\mathbf{z}_i\| \le \beta_z$, $\mathbf{Az}_i = c_i\mathbf{t} + \mathbf{w}_i$

Figure 2: Proof of Knowledge for $\bar{\mathbf{r}}$, with $\|\bar{\mathbf{r}}\| \le 2\beta_z$ satisfying $\mathbf{A}\bar{\mathbf{r}} = 2\mathbf{t}$. $C_x = \{0, X^i : 0 \le i < 2d\}$

Prover		**Verifier**
$\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix} \in R_q^{(k+1)\times(k+2)}, \mathbf{r}, \mathbf{L}, \mathbf{u}, \begin{bmatrix}0\\ \mathbf{m}\end{bmatrix}$ s.t. $\mathbf{m} \in R_q^k$, with $\mathbf{m}_i \in \mathbb{Z}_q$, $\mathbf{Lm} = \mathbf{u}$, $\begin{bmatrix}t_A\\ \mathbf{t}_B\end{bmatrix} = \begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}\mathbf{r} + \begin{bmatrix}0\\ \mathbf{m}\end{bmatrix}, \xi > \max_{c\in C} 11 \cdot \sqrt{3} \cdot \|c\mathbf{r}\|$		$\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}, \begin{bmatrix}t_A\\ \mathbf{t}_B\end{bmatrix}, \mathbf{L}, \mathbf{u}, \beta_z = \xi \cdot \sqrt{2(k+2)d}$
$\mathbf{y}, \mathbf{y}_1, \mathbf{y}_2 \leftarrow D_\xi^{k+2}$		
$w := \mathbf{Ay}, w_1 := \sigma_{-1}^{-1}(\mathbf{A})\mathbf{y}_1, w_2 := \sigma_5^{-1}(\mathbf{A})\mathbf{y}_2$		
$\mathbf{v}_1 := \mathbf{By} - \sigma_{-1}^{-1}(\mathbf{B})\mathbf{y}_1, \mathbf{v}_2 := \mathbf{By} - \sigma_5^{-1}(\mathbf{B})\mathbf{y}_2$		
$\mathbf{v}_L := \mathbf{LBy}$		
	$\xrightarrow{w, w_1, w_2, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_L}$	
	$\xleftarrow{c}$	$c \leftarrow C$
$\mathbf{z} := c\mathbf{r} + \mathbf{y}$,		
$\mathbf{z}_1 := c\sigma_{-1}^{-1}(\mathbf{r}) + \mathbf{y}_1, \mathbf{z}_2 := c\sigma_5^{-1}(\mathbf{r}) + \mathbf{y}_2$		
if $\mathtt{rej}([\mathbf{z} \mid \mathbf{z}_1 \mid \mathbf{z}_2], [c\mathbf{r} \mid c\sigma_{-1}^{-1}(\mathbf{r}) \mid c\sigma_5^{-1}(\mathbf{r})], \xi) = 1$, abort		
	$\xrightarrow{\mathbf{z}, \mathbf{z}_1, \mathbf{z}_2}$	
		Accept iff:
		$\|\mathbf{z}\|, \|\mathbf{z}_1\|, \|\mathbf{z}_2\| \le \beta_z$,
		$\mathbf{Az} = ct_A + w$,
		$\sigma_{-1}^{-1}(\mathbf{A})\mathbf{z}_1 = c\sigma_{-1}^{-1}(t_A) + w_1$,
		$\sigma_5^{-1}(\mathbf{A})\mathbf{z}_2 = c\sigma_5^{-1}(t_A) + w_2$,
		$\mathbf{Bz} - \sigma_{-1}^{-1}(\mathbf{B})\mathbf{z}_1 = c(\mathbf{t}_B - \sigma_{-1}^{-1}(\mathbf{t}_B)) + \mathbf{v}_1$
		$\mathbf{Bz} - \sigma_5^{-1}(\mathbf{B})\mathbf{z}_2 = c(\mathbf{t}_B - \sigma_5^{-1}(\mathbf{t}_B)) + \mathbf{v}_2$
		$\mathbf{LBz} = c(\mathbf{Lt}_B - \mathbf{u}) + \mathbf{v}_L$

Figure 3: Proof of Knowledge of $\bar{\mathbf{r}}, \bar{c}, \bar{\mathbf{m}}$, with $\|\bar{\mathbf{r}}\| \le 2\beta_z$ and $\bar{c} \in \bar{C}$, satisfying $\begin{bmatrix}\mathbf{A}\\ \mathbf{B}\end{bmatrix}\bar{\mathbf{r}} + \begin{bmatrix}0\\ \bar{c}\bar{\mathbf{m}}\end{bmatrix} = \bar{c}\begin{bmatrix}t_A\\ \mathbf{t}_B\end{bmatrix}$, $\mathbf{L}\bar{\mathbf{m}} = \mathbf{u}$, and $\bar{\mathbf{m}} \in \mathbb{Z}_q^k$. C and $\bar{C}$ are as in Figure 1.

Also note that combining (9) with (17) gives (8). Now, combining (14) and (17), we obtain

$$\mathbf{L}(\bar{c}\mathbf{t}_B - \bar{c}\bar{\mathbf{m}}) = \bar{c}(\mathbf{Lt}_B - \mathbf{u}),$$

which implies $\mathbf{L}\bar{\mathbf{m}} = \mathbf{u}$. □

3 THE DAA FRAMEWORK

Before presenting our new lattice-based DAA protocol, we recall how DAA works and what the desired security properties are.

3.1 DAA Functionality and Properties

In a DAA scheme, we have four main entities: a number of *trusted platform modules* (TPM), a number of *hosts*, an *issuer*, and a number of *verifiers*. A TPM and a host together form a platform which performs the **join** protocol with the issuer who decides if the platform is allowed to become a member. Once being a member, the TPM and host together can **sign** messages with respect to basenames *bsn*. If a platform signs with $bsn = \perp$ or a fresh basename, the signature must be anonymous and unlinkable to previous signatures. That is, any verifier can check that the signature stems from a legitimate platform via a deterministic **verify** algorithm, but the signature does not leak any information about the identity of the signer. Only when the platform signs repeatedly with the same basename $bsn \neq \perp$, it will be clear that the resulting signatures were created by the same platform, which can be publicly tested via a (deterministic) **link** algorithm. DAA also supports *key-based revocation*, i.e., it assumes the availability of a revocation list $\mathtt{RL}$ which contains the secret keys of rogue TPMs. Verification will be done with respect to such a revocation list and lets signatures of revoked TPMs fail.

Intuitively, DAA must satisfy the following high-level security and privacy properties. The formal security model for DAA exists in form of an ideal functionality $\mathcal{F}_{\mathsf{DAA}}^{l}$ in the UC framework [CDL16b].

Anonymity: An adversary that is given two signatures, w.r.t. two different basenames or $bsn = \perp$, cannot distinguish whether

Setup: Issuer Public Key: $ipk := (\mathbf{h}, \mathbf{b}, \mathbf{a}, u)$, Issuer Secret Key: $isk := \mathbf{R}$

Join: $(\mathsf{TPM}(tsk, ipk) \rightleftharpoons \mathsf{Host}(ipk)) \rightleftharpoons \mathsf{Issuer}(isk) : cred$

TPM Secret Key: $tsk := (e_1, e_2, sk)$
Host Credential: $cred := (\mathbf{s}, \tau)$ satisfying
$[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}] \cdot \mathbf{s} = u + a_1 e_1 + a_2 e_2$

Sign: $\mathsf{TPM}(tsk, ipk, \mu, bsn) \rightleftharpoons \mathsf{Host}(cred, ipk, \mu, bsn) : \sigma$

Signature: $\sigma := (\mathbf{t}, nym, d, \pi)$, where $\mathbf{t}$ is the commitment to τ, $d = H_{R_q}(bsn)$, $nym = de_1 + H_{R_3}(sk, bsn) \in R_q$, and π is the NIZK for equations (26)-(29) including μ in the Fiat-Shamir hash.

Verify: $\mathsf{Verify}(ipk, \mu, bsn, \sigma, \mathsf{RL}) \to (0, 1)$

Output 1 if π is correct, $d = H_{R_q}(bsn)$ (for $bsn \neq \perp$), and $\forall e_1 \in \mathsf{RL} : \|2(nym - de_1)\|$ is not small.

Link: $\mathsf{Link}(ipk, \mu_1, \mu_2, bsn, \sigma_1, \sigma_2) \to (0, 1)$

Output 1 if both signatures are valid and $2(nym_1 - nym_2)$ is small.

Figure 4: Overview of our DAA protocol

both signatures were created by one or two different honest platforms.

(One-More) Unforgeability: When the issuer is honest, an adversary controlling n TPMs can create at most n unlinkable signatures for the same basename $bsn \neq \perp$.

Non-Frameability: No adversary can create signatures on a message m w.r.t. basename bsn that links to a signature created by an honest platform, when this honest platform never signed m w.r.t. bsn.

Anonymity and non-frameability must hold even when the issuer is corrupt.

For the sake of simplicity, we present our DAA scheme in an algorithmic manner and discuss the additional UC-specific wrappers and inputs that are needed to satisfy $\mathcal{F}_{\mathsf{DAA}}^{l}$ in Section 4.

3.2 Protocol Description

Let q be a prime and the ring $R_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. We will denote by R_α for some positive integer α a subset of R_q with coefficients in the range between $-\alpha$ and α. For some domain D, we will denote H_D to be a domain extension function (e.g. SHAKE) that takes an element from $\{0,1\}^*$ and maps onto D.

The issuer secret key is a matrix $\mathbf{R} \leftarrow R_1^{2\times 2}$, and his public key is a uniformly-random polynomial $h \leftarrow R_q$ and a vector $\mathbf{b} = [h\ 1] \cdot \mathbf{R}$. By the Ring-LWE assumption, $(h, \mathbf{b})$ is indistinguishable from uniform. For convenience, we will write the public matrix associated to the issuer as

$$[\mathbf{h} \mid \mathbf{b}] \in R_q^4 \tag{20}$$

where $\mathbf{h} = [h\ 1]$. We will assume that when the issuer registers his public key with the CA, he also proves the knowledge of his secret key $\mathbf{R}$ in an extractable manner. Because the efficiency of this step is not very important, it can be performed using a standard Stern-type scheme (e.g [LNSW13]).

If $\mathbf{g} = [1\ \sqrt{q}] \in R_q^2$, then using the Micciancio-Peikert inversion algorithm, for any nonzero $\tau \in \mathbb{Z}_q$ and $u \in R_q$, it is possible to use the trapdoor $\mathbf{R}$ to find a short, Gaussian distributed, vector $\mathbf{s} \in R_q^4$ satisfying $[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}] \cdot \mathbf{s} = u$. The issuer will also keep state of one integer $tag\ \tau \in \mathbb{Z}_q$. He will initialize $\tau = 1$ and increment it by one with every new join. Since the prime q in our scheme will be somewhat large (around 2^{70}), every join procedure will have a unique tag – this is crucial for security.

We also define $\mathbf{a} = [a_1\ a_2] \leftarrow R_q^2$ and $u \leftarrow R_q$ to be random public parameters.

Join Procedure. A TPM's secret consists of a polynomial vector $\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \leftarrow R_3^2$ and a secret key $sk \in \{0,1\}^{256}$.[5] The TPM computes

$$u_1 = \mathbf{a} \cdot \mathbf{e} = a_1 e_1 + a_2 e_2 \tag{21}$$

and sends u_1 along with a proof of knowledge π_1 of short $\mathbf{e}$ satisfying (21). Since the TPM will do the join only once, it's not important for this proof to be very efficient, and so it can be done using a zero-knowledge proof system that proves *exact* knowledge of $\mathbf{e}$. The issuer, upon receiving u_1, π_1, will check the proof and then use the Micciancio-Peikert sampling algorithm to compute an $\mathbf{s}$ with small norm satisfying

$$[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}] \cdot \mathbf{s} = u + u_1, \tag{22}$$

where τ is a fresh tag which the issuer tracks as described above. The issuer sends τ and $\mathbf{s}$ to the TPM as the credential. The TPM keeps his secret $\mathbf{e}$ and sk, while the host stores the credential $\mathbf{s}$ and τ.

For DAA it is crucial that only legitimate TPMs can join in a controlled manner, which is handled via endorsement keys which are preinstalled on each TPM and their public keys are known to the issuer. We use the same abstraction as Camenisch et al. [CDL16b] and simply assume an authenticated channel $\mathcal{F}_{\mathsf{auth}}^{*}$ between the TPM (via the host) and the issuer. That is, during join the issuer learns the TPM identity $\mathcal{M}_i$ in an authenticated manner and should only proceed if $\mathcal{M}_i$ is a legitimate TPM. The issuer should also keep track of already joined TPMs, and ensure that each can join at most once.

Signing. To sign a message μ with respect to basename bsn (if the basename is $\perp$, the TPM picks a random bsn from the domain of all basenames, but doesn't reveal it) the TPM creates a value $d = H_{R_q}(bsn)$ and an error polynomial $e' = H_{R_3}(sk, bsn)$ and outputs the pseudonym $nym = de_1 + e' \in R_q$. Notice that d is (and needs to be) publicly computable, while e' is only computable by the TPM. The reason that the e' is generated deterministically based on bsn (and sk) rather than just chosen arbitrarily at random is that the TPM might be asked to create a pseudonym with respect to the same basename multiple times, and it would be insecure to send $de_1 + e'_1, \ldots, de_1 + e'_k$ for different e'_k. Thus, for the TPM's safety, the same basename should lead to the same pseudonym.

At this point, the TPM and the Host know short $\mathbf{e}, \mathbf{s}, e'$ and $\tau \in \mathbb{Z}_q$ satisfying

$$nym = de_1 + e' \tag{23}$$

$$[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}] \cdot \mathbf{s} = u + \mathbf{a} \cdot \mathbf{e}. \tag{24}$$

Ideally, the signature would consist of nym as well as a proof of knowledge of short $\mathbf{e}, \mathbf{s}, e'$ and $\tau \in \mathbb{Z}_q$ satisfying the above two

[5] The reason that we choose $\mathbf{e}$ to have coefficients in a range larger than $\{-1, 0, 1\}$ is because the TPM will give out a lot of Ring-LWE samples with this secret and so the space of the secrets needs to be a little larger to avoid the Arora-Ge attack.

equations. The main problem with creating the above proof is that it's unclear how to efficiently keep secret the τ inside the matrix $[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}]$ and, even ignoring the τ, giving an exact proof of $r, e, \mathbf{s}, r'$ is very costly in terms of proof size (at least dozens of megabytes). We instead use the techniques from Figures 1 and 2 to prove the above equations approximately as well as the proof of automorphism stability from 3 to prove that $\tau \in \mathbb{Z}_q$.

The Host commits to τ and $\tau\sqrt{q}$ using the commitment scheme from Section 2.5 as

$$\mathbf{C}\mathbf{r} + \begin{bmatrix} 0 \\ \tau \\ \tau\sqrt{q} \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}. \tag{25}$$

Define $\mathbf{C}_1$ and $\mathbf{C}_2$ to be the second and the third row of the matrix $\mathbf{C}$ in (6) (i.e. the rows corresponding to the commitments of messages m_1 and m_2 in (7). The Host and the TPM then jointly give a zero-knowledge proof π (using the message μ inside the random oracle in the Fiat-Shamir transformation) of small-normed $\bar{\mathbf{r}}, \bar{\mathbf{s}}, \bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \bar{\mathbf{e}}$, and $c \in \bar{C}$ satisfying:

$$\mathbf{C}\bar{\mathbf{r}} + \bar{c}\begin{bmatrix} 0 \\ \tau \\ \tau\sqrt{q} \end{bmatrix} = \bar{c}\begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix} \text{ and } \tau \in \mathbb{Z}_q \tag{26}$$

$$[\mathbf{h} \mid \mathbf{b} + [t_1\ t_2]]\bar{\mathbf{s}} - \mathbf{C}_1\bar{\mathbf{v}}_1 - \mathbf{C}_2\bar{\mathbf{v}}_2 - \mathbf{a}\cdot\bar{\mathbf{e}} = \bar{c}u \tag{27}$$

$$d\bar{e}_1 + \bar{e}' = \bar{c}nym \tag{28}$$

$$d\hat{e}_1 + \hat{e}' = 2nym \tag{29}$$

Equations (26), (27), and (28) are proved simultaneously (to ensure that the values of $\bar{c}$ and $\bar{e}_1$ are consistent throughout) and jointly by the TPM and the Host. In particular, (26) is proved using the "automorphism stability" proof from Figure 3 to ensure that $\tau \in \mathbb{Z}_q$, while the other two equations are proved using the standard "Fiat-Shamir with Aborts" technique using Gaussian sampling from Figure 1. The TPM needs to additionally prove (29) because having a $\bar{c}$ in front of the nym is not sufficient for linking since one would actually have to know $\mathbf{c}$ in order to perform the linking operation. This proof (which is less compact than the one in Figure 1 is done via the protocol in Figure 2.

While it's obvious that the fact that the TPM and Host satisfying (23) and (25) allows them to prove (26),(28), and (29), the validity of (27) is a little less straight-forward. But observe that replacing $\tau\mathbf{g} = [\tau\ \tau\sqrt{q}]$ in (24) with $\tau = t_1 - \mathbf{C}_1\mathbf{r}$ and $\sqrt{q}\tau = t_2 - \mathbf{C}_2\mathbf{r}$ from (25) gives us the equation

$$[\mathbf{h} \mid b_1 + t_1 \mid b_2 + t_2]\begin{bmatrix} \mathbf{s}_0 \\ s_1 \\ s_2 \end{bmatrix} - s_1\mathbf{C}_1\cdot\mathbf{r} - s_2\mathbf{C}_2\cdot\mathbf{r} - \mathbf{a}\cdot\mathbf{e} = u,$$

where we have conveniently rewritten $\mathbf{b} = [b_1\ b_2]$ and the vector $\mathbf{s} = \begin{bmatrix} \mathbf{s}_0 \\ s_1 \\ s_2 \end{bmatrix}$ where $\mathbf{s}_0$ is multiplied with $\mathbf{h}$, s_1 is multiplied by $b_1 + t_1$ and s_2 is multiplied by $b_2 + t_2$. Since s_1, s_2 and $\mathbf{r}$ have small coefficients, it's now evident that one can give a proof of the equation in (27) using the protocol in Figure 1.

The goal of obtaining proofs of (26) and (27) is to combine them into one proof as follows: by multiplying (27) by $\bar{c}$ and substituting $\bar{c}t_i$ with $\mathbf{C}_i\bar{r} + \bar{c}\tau$, we obtain

$$[\mathbf{h} \mid \bar{c}b_1 + \mathbf{C}_1\bar{\mathbf{r}} + \bar{c}\tau \mid \bar{c}b_2 + \mathbf{C}_2\bar{\mathbf{r}} + \bar{c}\sqrt{q}\tau]\begin{bmatrix} \bar{c}\bar{\mathbf{s}}_0 \\ \bar{s}_1 \\ \bar{s}_2 \end{bmatrix} - \mathbf{C}_1\bar{c}\bar{\mathbf{v}}_1 - \mathbf{C}_2\bar{c}\bar{\mathbf{v}}_2 - \mathbf{a}\cdot\bar{\mathbf{e}}\bar{c} = \bar{c}^2u, \tag{30}$$

which can be rewritten as

$$\bar{c}[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}]\bar{\mathbf{s}} = \bar{c}^2u + \mathbf{a}'\cdot\bar{\mathbf{w}}, \tag{31}$$

where $\mathbf{a}' = [\mathbf{a}\ \mathbf{C}_1\ \mathbf{C}_2]$ and $\bar{\mathbf{w}} = \begin{bmatrix} \bar{\mathbf{e}}\bar{c} \\ \bar{c}\bar{\mathbf{v}}_1 - \bar{\mathbf{r}} \\ \bar{c}\bar{\mathbf{v}}_2 - \bar{\mathbf{r}} \end{bmatrix}$.

The final signature output by the host is t_0, t_1, t_2, nym, d together with the proof π. Since $\mathbf{C}$ is a 3×4 matrix, the non-interactive proof of (26) consists of $4 \cdot 3 = 12$ polynomials (i.e. the size of the response in Figure 3. In the proof of (27), $\bar{\mathbf{s}}$ and $\bar{\mathbf{v}}_i$ each consist of 4 polynomials, while $\bar{\mathbf{e}}$ consists of 2, for a total of 14. The proof of (28) does not include any extra polynomials because one does not need to send anything corresponding to $\bar{e}'$ if one does an approximate check of the linear equation in the verification of Figure 1 (as in e.g. [BG14, DKL$^+$18]). To prove (29), one needs to set ℓ in Figure 2 to around 12, and so the output will be 12 polynomials (since one again does not need to output anything related to $\hat{e}'$). One should note that the polynomials corresponding to the $\bar{e}_1$ and $\hat{e}_1$ have rather small coefficients, so they will be less costly to output than the others. For a rough estimate, we ignore this savings. So the total number of elements of R_q in the proof π is 40, which makes the total number of polynomials in the signature 45.

Verification: The verifier, having message μ, basename bsn, signature $\sigma = (nym, d, \pi)$, and issuer public key ipk checks the proof π. If $bsn \neq \perp$, it also checks that $d = H_{R_q}(bsn)$. Further, DAA requires verification to be done with respect to a revocation list RL which contains all the rogue TPM's secret keys. Thus, for every $e_1 \in \mathsf{RL}$, the verifier checks that $\|2(nym - de_1)\|$ is not small. If all checks pass, it outputs 1 and 0 otherwise.

Linking. The link algorithm on input two signatures (μ_1, bsn, σ_1) and (μ_2, bsn, σ_2) for the same basename bsn checks whether both pseudonyms nym_1, nym_2 (which are part of the signatures) match. Given pseudonyms nym_1 and nym_2 for the same basename bsn, we will say that they are linked to the same TPM if $2(nym_1 - nym_2)$ is a polynomial in R_q with small norm.

To prove linkability, note that from (29) we have

$$2(nym_1 - nym_2) = d(\bar{e}_{1,1} - \bar{e}_{1,2}) + (\bar{e}'_1 - \bar{e}'_2),$$

where the left side has small norm and all the coefficients except for d on the right side also have small norm. This implies that there exist polynomials f_1, f_2 such that $df_1 + f_2 = 0$. One can show that, for a random choice of d, the probability that such non-zero f_1, f_2 exist is close to 0 (c.f. [KLS18, Proof of Lemma 4.8]). Therefore f_1 and f_2 must both be 0, which implies that $\bar{e}_{1,1} = \bar{e}_{1,2}$.

3.3 Security Intuition

The security intuition is that under the hardness of Ring-SIS and Ring-LWE, proving the knowledge of (31) implies that $\bar{\mathbf{e}} = \mathbf{e}\bar{c}$, where $\mathbf{e} = [e_1, e_2]$ is one of the TPM secret keys from (21) used during some Join session (Lemmas 3.1 and 3.2). For linking and revocation-checking, the presence of the $\bar{c}$ poses a problem because

this value can only be obtained by the extractor, but not the verifier. For this reason, we obtain proofs (28) and (29), which then imply that $\bar{e}_1 = 2e_1$ for some e_1 used during join. This then allows us to be certain that every *nym* is connected to some join session and, if given the secret key for the session, be able to link the *nym* to that secret key.

Our first lemma shows that one can construct a public key and a sampling algorithm, which are indistinguishable under certain computational assumptions from the real public key and sampling algorithm, that are conducive to extracting a solution to Ring-SIS via the procedure in Lemma 3.2.

Lemma 3.1. *Consider the public key*

$$[\mathbf{h} \mid \mathbf{b}], u, \mathbf{a} \tag{32}$$

generated as follows: Choose a random $\tau^* \in \mathbb{Z}_q, \mathbf{R} \in R_1^{2\times 2}, \mathbf{R}' \leftarrow R_{q^{1/4}}^{4\times 2}, \mathbf{s}_u \leftarrow D_\sigma$, *and set* $\mathbf{b} = \mathbf{h}\cdot\mathbf{R} - \tau^*\mathbf{g}, \mathbf{a} = [\mathbf{h} \mid \mathbf{hR}]\cdot\mathbf{R}'$, *and* $u = [\mathbf{h} \mid \mathbf{hR}]\cdot\mathbf{R}_u$.

Then there exists an efficient sampling algorithm producing $\mathbf{s} \sim D_\eta$ *such that no adversary can distinguish, based on the Ring-LWE and NTRU assumptions, between the real public key and the sampling algorithm, and the above public key and its sampling algorithm.*

Proof. The terms $\mathbf{a}$ and $\mathbf{u}$ in the real and the fake public key have the same distribution by the leftover hash lemma. Notice that the only difference between the real public key and the one in 32 is the extra $\tau^*\mathbf{g}$ term being added to $\mathbf{b}$. Through a series of games, we can show that, together with the sampling algorithm we describe below, the distribution of the public key and the sampling output is indistinguishable from the real public key and the real output.

Note that we cannot naively say that by the Ring-LWE assumption the real public key $[\mathbf{h} \mid \mathbf{hR}]$ is indistinguishable from the key $[\mathbf{h} \mid \mathbf{hR} - \tau^*\mathbf{g}]$ used in the reduction. The reason is that to use Ring-LWE, one needs to go through an intermediate distribution $[\mathbf{h} \mid \mathbf{b}]$ for a uniform $\mathbf{b}$, and in this case we do not know how to do any pre-image sampling since there is no trapdoor.

Instead, we will also be modifying the left side of the public key to create an NTRU trapdoor which we can use to do sampling when we change the right-hand side of the trapdoor to be uniform. The optimal way of selecting an NTRU trapdoor is outlined in [DLP14], and it allows for sampling pre-images with standard deviation less than the Micciancio-Peikert sampler (and so it can sample that standard deviation as well). We will also be making the decisional NTRU assumption that the NTRU public key f/g is indistinguishable from uniform. For f and g that are large with respect to q, as in [DLP14], this appears to be a reasonably safe assumption as slightly larger f and g result in truly random quotients [SS11]. We do not go into much detail here because the NTRU problem only appears in the proof and is not used anywhere in the scheme.

The reason that one must also modify the sampling algorithm is that in the real public key, one is always sampling an $\mathbf{s}$ satisfying $[\mathbf{h} \mid \mathbf{hR} + \tau\mathbf{g}]\mathbf{s} = u + \mathbf{a}\cdot\mathbf{e}$ where $\tau \neq 0$. This can be done with the Micciancio-Peikert sampling technique. With the public key in 32, on the other hand, if $\tau = \tau*$, then we need to sample an $\mathbf{s}$ satisfying $[\mathbf{h} \mid \mathbf{hR}]\mathbf{s} = u + \mathbf{a}\cdot\mathbf{e}$. Since the trapdoor $\mathbf{g}$ "vanishes", we can no longer use the aforementioned sampling algorithm. The way that this problem is generally overcome (e.g. in the original ABB selectively secure signature scheme) is that one sets the right-hand side of the equation $(u + \mathbf{a}\cdot e)$ to be $[\mathbf{h} \mid \mathbf{hR}]\mathbf{s}$. In our case, this is not possible because $\mathbf{e}$ is chosen by the adversary *after* he sees the public key. Our sampling algorithm for the special case of $\tau = \tau^*$ deals with this specific issue.

Game 1: $h \leftarrow R, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR}$. Sample using trapdoor $\mathbf{R}$.
Game 2: $h = f/g, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR}$. Sample using trapdoor $\mathbf{R}$. Indistinguishable by the NTRU assumption.
Game 3: $h = f/g, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR}$. Sample using NTRU trapdoor. Statistical Indistinguishability.
Game 4: $h = f/g, \mathbf{b} \leftarrow R^2$. Sample using NTRU trapdoor. Indistinguishable by Ring-LWE.
Game 5: $h = f/g, \mathbf{b}' \leftarrow R^2, \mathbf{b} = \mathbf{b}' - \tau^*\mathbf{g}$. Sample using NTRU trapdoor. Statistical indistinguishability.
Game 6: $h = f/g, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR} - \tau^*\mathbf{g}$. Sample using NTRU trapdoor. Indistinguishable by Ring-LWE.
Game 7: $h = f/g, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR} - \tau^*\mathbf{g}$. Sample using trapdoor $\mathbf{R}$ or the special procedure for τ^*. Statistical Indistinguishability.
Game 8: $h \leftarrow R, \mathbf{R} \leftarrow R_1^{2\times 2}, \mathbf{b} = \mathbf{hR} - \tau^*\mathbf{g}$. Sample using trapdoor $\mathbf{R}$ or the special procedure for τ^*. Indistinguishability by NTRU.

Sampling. For all $\tau \neq \tau^*$ and any $\mathbf{e}$, we use the Micciancio-Peikert sampling algorithm to sample an $\mathbf{s}$ satisfying $[\mathbf{h} \mid \mathbf{hR} + (\tau - \tau^*)\mathbf{g}]\cdot\mathbf{s} = u + \mathbf{a}\cdot\mathbf{e}$. This is possible because $\tau - \tau^* \neq 0$ and so the trapdoor does not vanish.

When $\tau = \tau^*$, then our trapdoor vanishes and so we cannot use it to do pre-image sampling. When asked to sign a message $\mathbf{e}^*$ with tag τ^*, the authority computes

$$\mathbf{s}^* = (\mathbf{s}_u + \mathbf{R}'\mathbf{e}^*). \tag{33}$$

This is a valid signature because $[\mathbf{h} \mid \mathbf{hR}]\mathbf{s}^* = u + \mathbf{a}\cdot\mathbf{e}^*$, but its distribution is incorrect because it's a shifted gaussian. Nevertheless, if $\|\mathbf{R}'\mathbf{e}^*\| \ll \|\mathbf{s}_u\|$, then one can use rejection sampling to transform the shifted gaussian distribution $\mathbf{s}_u - \mathbf{R}'\mathbf{e}^*$ into one that has the same distribution as $\mathbf{s}_u$ – i.e. D_ξ. If a reject happens, we abort the whole reduction and start over. Otherwise, the signature $\mathbf{s}^*$ has exactly the right distribution D_ξ. Recall that the authority will only need to sign for tag τ^* at most once and so as long as rejects don't happen with overwhelming probability, the success of the reduction only goes down by a polynomial factor with respect to the success probability of the forger. In order for the rejection probability to not be overwhelming, we need that $\|\mathbf{R}'\mathbf{e}^*\| < \sqrt{4d}\|\mathbf{s}_u\|$ (see Lemma 2.1). This will always be the case in our scheme. □

The below lemma shows that that every $\bar{e}_1$ extracted from the signer in (28) must be equal to $e_1^*\bar{c}$ for some e_1^* that the authority created a credential on in the Join procedure.

Lemma 3.2. *Suppose the public key is as in Lemma 3.1. If there is an adversary that after choosing, for* $1 \le i \le \kappa$, $\mathbf{e}^{(i)}$ *and receiving* $\mathbf{s}^{(i)}$ *satisfying* $[\mathbf{h} \mid \mathbf{b} + \tau\mathbf{g}]\cdot\mathbf{s}^{(i)} = u + \mathbf{a}\cdot\mathbf{e}^{(i)}$ *is able to output a* $\tau \in \mathbb{Z}_q$, *an* $\bar{\mathbf{s}} \in R_q^4$ *with* $\|\bar{\mathbf{s}}\| \le \beta_{\bar{s}}$, *a* $\bar{c} \in \bar{C}$, *and an* $\bar{\mathbf{e}}$ *s.t.* $\forall\, i\, \bar{\mathbf{e}} \neq \bar{c}\mathbf{e}^{(i)}$, *satisfying*

$$\bar{c}[\mathbf{h} \mid (\mathbf{b} + \tau\mathbf{g})]\cdot\bar{\mathbf{s}} = \bar{c}^2 u + \mathbf{a}\cdot\bar{\mathbf{e}} + [\mathbf{C}_1 \mid \mathbf{C}_2]\cdot\begin{bmatrix}\bar{\mathbf{w}}_1\\ \bar{\mathbf{w}}_2\end{bmatrix}, \tag{34}$$

(where all the notation is as in (31)*) then there exists another algorithm that can solve a Ring-SIS instance* $[\mathbf{h} \mid \mathbf{C}_1 \mid \mathbf{C}_2]$ *with probability*

approximately $1/q$ smaller than the success probability of the adversary.

Proof. With probability $1/q$ the tag τ that the adversary forges on is τ^*. In other words, the adversary returns a $\bar{\mathbf{s}}, \bar{c}, \bar{\mathbf{w}}$ satisfying

$$\bar{c}[\mathbf{h} \mid \mathbf{b}']\bar{\mathbf{s}} = \bar{c}^2 u + \mathbf{a} \cdot \bar{\mathbf{e}}\bar{c} + \mathbf{C}_1\bar{\mathbf{w}}_1 + \mathbf{C}_2\bar{\mathbf{w}}_2, \tag{35}$$

where $\mathbf{b}' = \mathbf{hR}$.

From (33), we also have another equation

$$[\mathbf{h} \mid \mathbf{b}']\mathbf{s}^* = u + \mathbf{a} \cdot \mathbf{e}^*. \tag{36}$$

Multiplying (36) by $\bar{c}^2$ and subtracting from (35), and then plugging in $\mathbf{a} = [\mathbf{h}\ \mathbf{b}]\mathbf{R}'$, we obtain

$$[\mathbf{h} \mid \mathbf{b}'](\bar{c}\bar{\mathbf{s}} - \bar{c}^2\mathbf{s}^*) = [\mathbf{h} \mid \mathbf{b}']\mathbf{R}'(\bar{\mathbf{e}}\bar{c} - \mathbf{e}^*\bar{c}^2) + \mathbf{C}_1\bar{\mathbf{w}}_1 + \mathbf{C}_2\bar{\mathbf{w}}_2 \tag{37}$$

Because everything being multiplied by $\mathbf{h}, \mathbf{b}', \mathbf{C}_1$, and $\mathbf{C}_2$ has small coefficients, this will be a SIS solution to $[\mathbf{h} \mid \mathbf{b}' \mid \mathbf{C}_1\ \mathbf{C}_2]$ (and therefore to $[\mathbf{h} \mid \mathbf{C}_1 \mid \mathbf{C}_2]$ since $\mathbf{b}' = \mathbf{hR}$) unless all the multiplicands are 0. Because the adversary cannot predict the exact value of $\mathbf{R}'$ (because the entropy of each column of $\mathbf{R}'$ is larger than of each column of $\mathbf{a}$), in order to force the extractor to extract a zero-solution, the adversary would need $\bar{\mathbf{e}}\bar{c} = \mathbf{e}^*\bar{c}^2$. By the invertibility of $\bar{c}$, this implies that $\bar{\mathbf{e}} = \mathbf{e}^*\bar{c}$.

□

The below lemma proves that (28) and (29), together with the knowledge that $\bar{e}_1 = e_1^*\bar{c}$, imply that $\hat{e}_1 = 2e_1^*$.

Lemma 3.3. *If, for a randomly-chosen $d \in R_q$, the following equations hold:*

$$\bar{e}_1 = e_1^*\bar{c} \tag{38}$$

$$d\bar{e}_1 + \bar{e}' = \bar{c}nym \tag{39}$$

$$d\hat{e}_1 + \hat{e}' = 2nym \tag{40}$$

and $\|\bar{e}_1\|_\infty, \|e_1\|_\infty, \|\hat{e}_1\|_\infty, \|\hat{e}'\|_\infty \ll \sqrt{q/2}$, then with probability close to 1 (over the choice of d) $\hat{e}_1 = 2e_1^*$.

Proof. Multiplying (39) by 2 and (40) by $\bar{c}$, subtracting, and plugging in (38), we obtain

$$d(2e_1^*\bar{c} - \hat{e}_1\bar{c}) + (2\bar{e}' - \hat{e}'\bar{c}) = 0.$$

Because all the variables except d in the above equation are $\ll \sqrt{q/2}$, by [KLS18, Lemma 4.8] we know that it's highly unlikely that a non-zero solution $du_1 + u_2 = 0$ exists for small, non-zero u_i. Therefore $2e_1^*\bar{c} = \hat{e}_1\bar{c}$, and because $\bar{c}$ is invertible, we get the claim in the lemma. □

4 SECURITY

We now show that our DAA protocol satisfies the desired DAA security guarantees captured through the ideal functionality $\mathcal{F}_{\mathsf{DAA}}^l$ [CDL16b]. Before presenting our proof sketch, we first discuss how the protocols and algorithms presented in Section 3 have to be "UC-fied".

4.1 UC Wrapper for our Protocol

To date, the only sound security notion for DAA is an ideal functionality in the Universal Composability framework. Describing protocols in the UC framework requires some extra care to include session identifiers, explicit party inputs in interactive protocols, as well as to reflect the abstract modeling of keys and secure channels. We start by describing the necessary sub-functionalities our protocol relies on, and then discuss how to map our protocols to the interfaces required by $\mathcal{F}_{\mathsf{DAA}}^l$.

Sub-Functionalities. We assume that a common reference string functionality $\mathcal{F}_{\mathsf{crs}}$ and a certificate authority functionality $\mathcal{F}_{\mathsf{ca}}$ are available to all parties. The later allows the issuer to register his public key, and $\mathcal{F}_{\mathsf{crs}}$ is used to provide all entities with the system parameters comprising of the random seed to generate the commitments, and $\mathbf{a}$ of the issuer's public key.

For the communication between the TPM and issuer (via the host) in the join protocol, we use the semi-authenticated channel $\mathcal{F}_{\mathsf{auth}}^*$ introduced in [CDL16b]. For all communication between a host and TPM we assume the secure message transmission functionality $\mathcal{F}_{\mathsf{smt}}$ (enabling authenticated and encrypted communication). In practice, $\mathcal{F}_{\mathsf{smt}}$ is naturally guaranteed by the physical proximity of the host and TPM forming the platform, i.e., if both are honest an adversary can neither alter nor read their internal communication.

In the description of the protocol, we assume that parties call $\mathcal{F}_{\mathsf{crs}}$ and $\mathcal{F}_{\mathsf{ca}}$ to retrieve the necessary key material whenever they use a public key of another party. Further, if any of the checks in the protocol fails, the protocol ends with a failure message $\perp$.

The protocol also outputs $\perp$ whenever a party receives an input or message it does not expect (e.g., protocol messages arriving in the wrong order.)

$\mathcal{F}_{\mathsf{DAA}}^l$ Interfaces. The $\mathcal{F}_{\mathsf{DAA}}^l$ functionality considers an issuer $\mathcal{I}$ and the platform consisting of a TPM $\mathcal{M}_i$ and a host $\mathcal{H}_j$. In UC, different instances of a protocol are separated through unique session identifiers $sid = (\mathcal{I}, sid')$. In the real-world these are mapped to the issuer public key, and all parties use the *sid* to link their stored key material to the particular issuer.

Setup: $\mathcal{I}$ upon input (SETUP, *sid*) generates his key pair (*ipk*, *isk*). It registers the public key (*sid*, *ipk*) at $\mathcal{F}_{\mathsf{ca}}$, stores the secret key as (*sid*, *isk*) and ends with output (SETUPDONE, *sid*).

Join: To distinguish several join sessions that might run in parallel, we use a unique sub-session identifier *jsid* that is given as additional input to all parties. The join protocol starts when $\mathcal{H}_j$ receives the input (JOIN, *sid*, *jsid*, $\mathcal{M}_i$) upon which it triggers $\mathcal{M}_i$ to generate u_1 along with the proof π_1, and send it via $\mathcal{F}_{\mathsf{auth}}^*$ to $\mathcal{I}$. When $\mathcal{I}$ receives the message it outputs (JOINPROCEED, *sid*, *jsid*, $\mathcal{M}_i$). The join session is completed when the issuer receives an input telling him to proceed with join session *jsid*, upon which it returns $cred = (\mathbf{s}, \tau)$. This explicit interaction with the issuer allows the issuer to perform some additional check to decide whether $\mathcal{M}_i$ is allowed to join. The host stores the credential as (*sid*, $\mathcal{M}_i$, *cred*) and the TPM stores its secret key as (*sid*, $\mathcal{H}_j$, *tsk*) i.e., both "remember" with whom they joined with. The join ends with $\mathcal{M}_i$ outputting (JOINED, *sid*, *jsid*).

Sign: Signing is a protocol run between a TPM $\mathcal{M}_i$ and a host $\mathcal{H}_j$.

Again, we use a unique sub-session identifier *ssid* to allow for multiple sign sessions and unique identification of the particular session in the UC interfaces. The host $\mathcal{H}_j$ upon input (SIGN, *sid*, *ssid*, $\mathcal{M}_i$, μ, *bsn*), retrieves its join record (*sid*, $\mathcal{M}_i$, *cred*) and aborts if no such record is found. It sends (*ssid*, μ, *bsn*) to the TPM which then checks that a key record (*sid*, $\mathcal{H}_j$, *tsk*) exists and outputs (SIGNPROCEED, *sid*, *ssid*, μ, *bsn*). The signature is completed when $\mathcal{M}_i$ receives the input (SIGNPROCEED, *sid*, *ssid*), upon which it computes the SPK and *nym*. The explicit input from the TPM is necessary to ensure that the TPM in fact "approved" the attestation of μ and *bsn*. Finally, the host outputs the jointly computed signature as (SIGNATURE, *sid*, *ssid*, σ).

Verify and Link: Here both algorithms are simply re-labeled as UC interfaces with the *ipk* being replaced with *sid*, i.e. both algorithms are made available through interfaces (VERIFY, *sid*, μ, *bsn*, σ, RL) and (LINK, *sid*, σ, μ, σ', μ', *bsn*) respectively.

4.2 Proof Sketch

Theorem 4.1. *The protocol* Π_{daa} *presented in Section 3 securely realizes* $\mathcal{F}_{\mathsf{DAA}}^{l}$ *[CDL16b] in the* $(\mathcal{F}_{\mathsf{auth}}^{*}, \mathcal{F}_{\mathsf{ca}}, \mathcal{F}_{\mathsf{smt}}, \mathcal{F}_{\mathsf{crs}})$*-hybrid and random oracle model under static corruptions, if the Ring-LWE, Ring-SIS, and the NTRU assumption hold.*

To show that no environment $\mathcal{E}$ can distinguish the real world, in which it is working with Π_{daa} and adversary $\mathcal{A}$, from the ideal world, in which it uses $\mathcal{F}_{\mathsf{DAA}}^{l}$ with simulator $\mathcal{S}$, we use a sequence of games. We start with the real world protocol execution. In the next game we construct one entity C that runs the real world protocol for all honest parties. Then we split C into two pieces, a functionality $\mathcal{F}$ and a simulator $\mathcal{S}$, where $\mathcal{F}$ receives all inputs from honest parties and sends the outputs to honest parties. We start with a useless functionality, and gradually change $\mathcal{F}$ and update $\mathcal{S}$ accordingly, to end up with the full $\mathcal{F}_{\mathsf{DAA}}^{l}$ and a satisfying simulator.

The proof closely follows the structure of the UC proof by Camenisch et al. in [CDL16b], with the crucial steps occurring in Game 7, where the signatures of honest platforms are replaced by signatures on "dummy" keys (guaranteeing *anonymity*), and Games 12–15 where we let the functionality enforce the expected *unforgeability* and *non-frameability* properties. For unforgeability we rely on the security of signatures created by the issuer, which we have shown to hold in Lemma 3.1-3.3.

Game 1: This is the real world protocol.

Game 2: The challenger C now receives all inputs and simulates the real world protocol for honest parties. Since C gets all inputs, it can simply run the real world protocol. It also simulates all hybrid functionalities, but does so honestly, so $\mathcal{E}$ does not see any difference. By construction, this is equivalent to the previous game.

Game 3: We now split C into a "dummy functionality" $\mathcal{F}$ and simulator $\mathcal{S}$. $\mathcal{F}$ receives all inputs, and simply forwards them to $\mathcal{S}$. $\mathcal{S}$ simulates the real world protocol and sends the outputs it generates to $\mathcal{F}$, who then outputs them to $\mathcal{E}$. This game only restructures the previous game.

Game 4: In this game we let our intermediate $\mathcal{F}$ handle the setup related interfaces using the procedure specified in $\mathcal{F}_{\mathsf{DAA}}^{l}$. Consequently, $\mathcal{F}$ expects to receive the algorithms (ukgen, sig, ver, link, identify) from the simulator. For ukgen, ver, and link, $\mathcal{S}$ can simply provide the algorithms from the real-world protocol, where it omits the revocation check from ver. The sig algorithm will be a combination of the join and sign procedure though, as it will be used to create anonymous signatures for honest platforms for which it uses a fresh TPM key whenever the platform signs w.r.t. a new basename. Thus, to internally create signatures via sig, the algorithm must first create a valid membership credential for the freshly chosen *tsk* and then sign with this new credential. So sig must contain the issuer's private key, which the simulator $\mathcal{S}$ has to be able to get.

When $\mathcal{I}$ is honest, $\mathcal{S}$ is running the issuer, i.e., it knows its secret key and sets the sig algorithm accordingly.

When $\mathcal{I}$ is corrupt, $\mathcal{S}$ starts the simulation when the issuer registers his key with $\mathcal{F}_{\mathsf{ca}}$ that is controlled by the simulator. Since the public key comes with a proof of knowledge of the issuer's secret key, $\mathcal{S}$ can extract the secret key from there and define the sig algorithm accordingly. By the simulation soundness of the proof system, this game hop is indistinguishable for the adversary.

Finally, for identify which is used to check whether a signature (σ, μ, bsn) belongs to a certain *tsk*, we use roughly the same procedure as for revocation checks. That is, the algorithm parses $\sigma = (nym, d, bsn)$, $tsk = (e_1, *)$, and checks that $\|2(nym - de_1)\|$ is small. If so it outputs 1, and 0 otherwise. Recall that we compute pseudonyms for random d when $bsn = \perp$, so this check works for all cases.

Game 5: $\mathcal{F}$ now handles the verify and link queries using the provided algorithms ver and link from the previous game, rather than forwarding the queries to $\mathcal{S}$. We do not let $\mathcal{F}$ perform the additional checks (Checks (ix) – (xvi)) done by $\mathcal{F}_{\mathsf{DAA}}^{l}$, though, but add these only later. For Check (xii), $\mathcal{F}$ rejects a signature when a matching $tsk' \in \mathsf{RL}$ is found, but does not exclude honest TPMs from this check yet.

Because verify and link do not involve network traffic, the simulator does not have to simulate traffic either, we must only make sure the outputs do not change. $\mathcal{F}$ executes the algorithms that $\mathcal{S}$ supplied, and $\mathcal{S}$ supplied them in such a way that they are equivalent to the real world algorithms, so the outcome will clearly be equivalent.

Game 6: In this step we change $\mathcal{F}$ to also handle the join-related interfaces, meaning it will receive the inputs and generate the outputs. We let $\mathcal{F}$ run the same procedure as $\mathcal{F}_{\mathsf{DAA}}^{l}$, but again omit the additional checks (Checks (ii)–(iv)).

In the final join interface JOINCOMPLETE, the simulator has to provide the secret key *tsk* of the TPM. When the TPM is honest, $\mathcal{S}$ knows the key anyway and uses it towards $\mathcal{F}$. If the TPM is corrupt and either the issuer or host is honest, $\mathcal{S}$ extracts the vector e_1 from the proof π_1 that it receives in the role of the honest $\mathcal{I}$ or $\mathcal{H}_j$ and sets $tsk \leftarrow (e_1, \perp)$. Note that we do not extract nor set the part *sk* of the TPM's secret key. This has no impact though, as *tsk* will only be used for internal checks by identify for which only e_1 is used.

Finally, note that $\mathcal{F}$ sets $tsk \leftarrow \perp$ when both the TPM and host are honest. However, this has no impact yet, as the signatures are still created by the simulator and the verify and link interfaces of $\mathcal{F}$ do not run the additional checks that make use of the internally stored records and keys.

Overall, this game hop is indistinguishable by the simulation soundness of the SPK π_1.

Game 7: We now transform $\mathcal{F}$ such that it internally handles the signing queries of *honest platforms* instead of merely forwarding them to $\mathcal{S}$. Thus, this game hop proves the **anonymity** of our DAA scheme.

Again, $\mathcal{F}$ uses the sign interfaces from $\mathcal{F}_{\text{DAA}}^{l}$, with the difference that it does not perform the Check (v) – (vii) which we only add in a later game.

When both the TPM and the host are honest, $\mathcal{F}$ creates the signatures internally in an unlinkable way: It chooses a new tsk per basename and TPM, or per signature when $bsn = \perp$ and then runs the sig algorithm for that fresh key. As described earlier, sig starts by internally "issuing" a membership credential on tsk using the issuer's secret key that is included in sig. $\mathcal{F}$ keeps the internally chosen keys $\langle \mathcal{M}_i, bsn, tsk \rangle$ in a list `DomainKeys` to ensure consistency if a TPM wishes to reuse the basename.

This change is indistinguishable under the Ring-LWE assumption, the reduction can be done in a straightforward manner, using a hybrid argument to to replace the signatures one-by-one.

Game 8: We change $\mathcal{F}$ such that it no longer informs $\mathcal{S}$ which message and basename are being signed. Thus, when the TPM and host are both honest, $\mathcal{S}$ does not learn μ, bsn but only its leakage $l(\mu, bsn)$. Recall, that signatures for honest platforms are generated by the functionality now, so $\mathcal{S}$ merely as to mimic communication between the honest TPM and host.

Game 9: We now add the constraint that when $\mathcal{I}$ is honest, $\mathcal{F}$ only allows platforms that joined to sign, which is checked via the list `Members`. Note that for our simulation we only care about platforms that are at least "partially" honest, i.e., the host and/or TPM are honest, as otherwise there is nothing to simulate. For such platforms, this check will not change the view of $\mathcal{E}$ using the simulator $\mathcal{S}$ from the previous game: in the real world, an honest host and TPM both check that they have a joined before signing. In the ideal world, $\mathcal{S}$ makes join queries towards $\mathcal{F}$ ensuring that the joined platforms (with honest entities) are in `Members`, and thus $\mathcal{F}$ still allows any signing that could take place in the real world.

Game 10: In this game we let $\mathcal{F}$ additionally check the validity of every new tsk that is generated or received in the join and sign interface.

If the TPM is corrupt, $\mathcal{F}$ checks that $\mathsf{CheckTskCorrupt}(tsk) = 1$ for the tsk that the simulator extracted from π_1 (Check (iv)). This check prevents the adversary from choosing different keys $tsk \neq tsk'$ that both fit to the same signature. By the It is easy to see that there exists only one secret e_1 satisfying the nym-part of the signature (also for the "random" pseudonyms when $bsn = \perp$). As there is only a single tsk for every valid signature where $\mathsf{identify}(\sigma, \mu, bsn, tsk) = 1$, this check will never fail.

For keys of honest TPMs, $\mathcal{F}$ verifies that $\mathsf{CheckTskHonest}(tsk) = 1$ whenever it receives or generates a new tsk (Check (iii) and (v)). With these checks we avoid the registration of keys for which matching signatures already exist. Since keys for honest TPMs are chosen uniformly at random from an exponentially large group and every signature has exactly one matching key, the chance that a signature under that key already exists is negligible.

Game 11: We now add the checks to $\mathcal{F}$ that $\mathcal{F}_{\text{DAA}}^{l}$ runs in the sign interfaces when internally generating signatures for honest platforms. After creating a signature, $\mathcal{F}$ checks whether the signature verifies and matches the right key (Check (vi) and (vii)). As $\mathcal{S}$ supplied proper algorithms and the signature scheme is complete, these checks will obviously always succeed.

$\mathcal{F}$ also checks with the help of its internal key records `Members` and `DomainKeys` that no one else already has a key which would match this newly generated signature (Check viii). As signatures match only a single TPM key and we choose keys of honest platforms at random from a large domain, this can happen only with negligible probability.

In the next four game hops, we let $\mathcal{F}$ perform the four additional checks that are done by $\mathcal{F}_{\text{DAA}}^{l}$ in the verification interface, i.e., we rely on $\mathcal{F}$ to enforce the desired **unforgeability** and **non-frameability** guarantees. We now show that this check does not change the verification outcome, as any signature that would previously pass will still pass.

Game 12: $\mathcal{F}$ now performs an additional check during verification, it checks whether it finds multiple tsk values matching this signature, and if so, it rejects the signature. It is easy to see that there is only tsk per signature for which identify will output 1, as there is only one e_1 that can lead to the pseudonym nym.

Game 13: When $\mathcal{I}$ is honest, $\mathcal{F}$ now only accepts signatures on tsk values that $\mathcal{I}$ has issued a membership credential on. Under the existential unforgeability of the membership credential, this check changes the verification outcome only with negligible probability. The unforgeability of our signature scheme used by the issuer to (blindly) sign the TPM's key is based on the Ring-SIS and Ring-LWE problem as shown in Lemma 3.1-3.3.

Game 14: $\mathcal{F}$ now prevents forging signatures using an honest TPM's tsk. If the environment can distinguish this game hop, i.e., it can create a valid signature σ for message μ and basename bsn that traces via identify to an honest TPM, but that TPM has never signed μ, bsn we can use this to break the Ring-LWE problem. Again, the reduction can be done in a straight-forward manner: We use the Ring-LWE challenge as u_1 for a randomly chosen honest TPM during the join protocol (with the possibly corrupt issuer). DAA Signature for this honest TPM are merely simulated. If we see a valid signature that we never created, we extract e_1 from there and use it to break the Ring-LWE problem.

Game 15: Check (xii) is added to $\mathcal{F}$, this ensures that honest TPMs are not being revoked. If an honest TPM is simulated by means of the Ring-LWE problem instance, if a proper key `RL` is found, it must be the secret key of the target instance. This is again equivalent to solving the Ring-LWE problem.

Game 16: We now let $\mathcal{F}$ perform all the additional checks $\mathcal{F}^{l}_{\text{DAA}}$ makes for link queries. The output of $\mathcal{F}$ based on these checks is still consistent with the output which the link algorithm would give: If there is a *tsk* that matches one signature but not the other, by soundness of π we have that the pseudonyms are not based on the same *tsk*, and thus must differ which results in link outputting 0. If there is a *tsk* that matches both signatures, by soundness of π we have that the pseudonyms are based on the same *tsk* and must be equal, resulting in link outputting 1.

Now $\mathcal{F}$ is equal to $\mathcal{F}^{l}_{\text{DAA}}$, concluding our proof sketch. □

Acknowledgements. This work is supported by the EU Horizon 2020 research and innovation program under grant agreement No 779391 (FutureTPM).

REFERENCES

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security, CCS 2004*. ACM, 2004.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT (1)*, volume 8873 of *LNCS*, 2014.

[BCL08] Ernie Brickell, Liqun Chen, and Jiangtao Li. A new direct anonymous attestation scheme from bilinear maps. In *Trusted Computing - Challenges and Applications,*, pages 166–178, 2008.

[BCL09] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *Int. J. Inf. Sec.*, 8(5):315–330, 2009.

[BDL+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *SCN*, volume 11035 of *LNCS*, 2018.

[Beu19] Ward Beullens. On sigma protocols with helper for mq and pkp, fishy signature schemes and more. ePrint Archive, Report 2019/490, 2019.

[BFG+13] David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 12(3):219–249, 2013.

[BG14] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014.

[BK17] Rachid El Bansarkhani and Ali El Kaafarani. Direct anonymous attestation from lattices. In *Cryptology ePrint Archive, Report 2017/1022*, 2017.

[BL07] Ernie Brickell and Jiangtao Li. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *ACM Workshop on Privacy in the Electronic Society WPES*, pages 21–30, 2007.

[BLS19] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO*, 2019.

[BN19] Carsten Baum and Ariel Nof. Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. ePrint Archive, Report 2019/532, 2019.

[BS15] Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In *Post-Quantum Cryptography*, 2015.

[CCD+17] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *IEEE Symposium on Security and Privacy, SP*, pages 901–920, 2017.

[CDE+] Jan Camenisch, Manu Drijvers, Alec Edgington, Anja Lehmann, Rolf Lindemann, and Rainer Urian. FIDO ECDAA algorithm, implementation draft. https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-ecdaa-algorithm-v1.1-id-20170202.html.

[CDL16a] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *Trust and Trustworthy Computing TRUST*, 2016.

[CDL16b] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In *Public-Key Cryptography - PKC 2016*, pages 234–264, 2016.

[CDL17] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation with subverted tpms. In *CRYPTO 2017*, pages 427–461, 2017.

[CPS10] Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. In *Smart Card Research and Advanced Application, CARDIS*, pages 223–237, 2010.

[CU15] Liqun Chen and Rainer Urian. DAA-A: direct anonymous attestation with attributes. In *Trust and Trustworthy Computing, TRUST*, pages 228–245, 2015.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT (2)*, 2014.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO*, pages 335–352, 2014.

[DP16] Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In *ISSAC*, pages 191–198. ACM, 2016.

[dPLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.

[GCH+19] Wen Gao, Liqun Chen, Yupu Hu, Christopher J. P. Newton, Baocang Wang, and Jiangshan Chen. Lattice-based deniable ring signatures. In *International Journal of Information Security 18(3):355–370*, 2019.

[GJL11] Ulrich Greveler, Benjamin Justus, and Dennis Loehr. Direct anonymous attestation: enhancing cloud service user privacy. In *OTM Confederated International Conf."On the Move to Meaningful Internet Systems"*, 2011.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[Int13] International Organization for Standardization. ISO/IEC 20008-2: Information technology - Security techniques - Anonymous digital signatures - Part 2: Mechanisms using a group public key, 2013.

[Int15] International Organization for Standardization. ISO/IEC 11889: Information technology - Trusted platform module library, 2015.

[KCB+19] Nada Kassem, Liqun Chen, Rachid El Bansarkhani, Ali El Kaafarani, Jan Camenisch, Patrick Hough, Paulo Sérgio Alves Martins, and Leonel Sous. More efficient, provably-secure direct anonymous attestation from lattices. In *Future Generation Computer Systems*, 2019.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In *EUROCRYPT (3)*, volume 10822 of *LNCS*, 2018.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389, 2008.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016*, 2016.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, volume 4052 of *LNCS*, 2006.

[LN17] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT (1)*, volume 10210 of *LNCS*, 2017.

[LNSW13] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC*, pages 107–124, 2013.

[LNW15] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *PKC*, 2015.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *LNCS*, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *EUROCRYPT*, pages 27–47, 2011.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21, 1993.

[Tru04] Trusted Computing Group. TPM main specification version 1.2, 2004.

[Tru14] Trusted Computing Group. Trusted platform module library specification, family "2.0", 2014.

[WCG+17] Jorden Whitefield, Liqun Chen, Thanassis Giannetsos, Steve Schneider, and Helen Treharne. Privacy-enhanced capabilities for vanets using direct anonymous attestation. In *Vehicular Networking Conference (VNC)*, 2017.

[XYZF14] Li Xi, Kang Yang, Zhenfeng Zhang, and Dengguo Feng. Daa-related apis in TPM 2.0 revisited. In *Trust and Trustworthy Computing TRUST*, 2014.

[YAZ+19] Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO*, 2019.