

PRIVÉ

Towards Privacy-Preserving Swarm Attestation

El Kassem, Nada; Hellemans, Wouter; Siachos, Ioannis; Dushku, Edlira; Vasileiadis, Stefanos; Karas, Dimitrios S. ; Chen, Liqun; Patsakis, Constantinos; Giannetsos, Thanassis

Published in:

Proceedings of the 22nd International Conference on Security and Cryptography

DOI (link to publication from Publisher):

[10.5220/0013629000003979](https://doi.org/10.5220/0013629000003979)

Creative Commons License

CC BY-NC-ND 4.0

Publication date:

2025

Document Version

Accepted author manuscript, peer reviewed version

[Link to publication from Aalborg University](#)

Citation for published version (APA):

El Kassem, N., Hellemans, W., Siachos, I., Dushku, E., Vasileiadis, S., Karas, D. S., Chen, L., Patsakis, C., & Giannetsos, T. (2025). PRIVÉ: Towards Privacy-Preserving Swarm Attestation. In S. De Capitani Di Vimercati, & P. Samarati (Eds.), *Proceedings of the 22nd International Conference on Security and Cryptography* (pp. 247-262) <https://doi.org/10.5220/0013629000003979>

General rights










Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy

If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

PRIVÉ: Towards Privacy-Preserving Swarm Attestation

Nada El Kassem¹^a, Wouter Hellemans²^b, Ioannis Siachos³^c, Edlira Dushku⁴^d, Stefanos Vasileiadis³^e, Dimitrios S. Karas³^f, Liqun Chen¹^g, Constantinos Patsakis⁸^h, Thanassis Giannetsos³ⁱ

¹ University of Surrey, Guildford, UK

² ES&S, COSIC, ESAT, KU Leuven, Leuven, Belgium

³ UBITECH Ltd., Athens, Greece

⁴ Aalborg University, Copenhagen, Denmark

⁸ University of Piraeus: Piraeus, Attiki, Greece

{nada.elkassem, liqun.chen}@surrey.ac.uk, wouter.hellemans@kuleuven.be, {isiachos, svasileiadis, dkarras, agiannetsos}@ubitech.eu, edu@es.aau.dk, kpatsak@unipi.gr

Keywords: Swarm Attestation, Privacy, Direct Anonymous Attestation, Remote Attestation, In-Vehicle Networks


Abstract: In modern large-scale systems comprising multiple heterogeneous devices, the introduction of swarm attestation schemes aims to alleviate the scalability and efficiency issues of traditional single-Prover and single-Verifier attestation. In this paper, we propose PRIVÉ, a privacy-preserving, scalable, and accountable swarm attestation scheme that addresses the limitations of existing solutions. Specifically, we eliminate the assumption of a trusted Verifier, which is not always applicable in real-world scenarios, as the need for the devices to share identifiable information with the Verifier may lead to the expansion of the attack landscape. To this end, we have designed an enhanced variant of the Direct Anonymous Attestation (DAA) protocol, offering traceability and linkability whenever needed. This enables PRIVÉ to achieve anonymous, privacy-preserving attestation while also providing the capability to trace a failed attestation back to the compromised device. To the best of our knowledge, this paper presents the first Universally Composable (UC) security model for swarm attestation accompanied by mathematical UC security proofs, as well as experimental benchmarking results that highlight the efficiency and scalability of the proposed scheme.

1 INTRODUCTION

In recent years, the exponential proliferation of low-cost embedded devices and the Internet of Things (IoT) has significantly contributed to developing innovative environments, particularly in advancing Intelligent Transportation Systems. Unfortunately, these devices represent natural and attractive malware attack targets despite many benefits. To ensure the correctness of the operational state of a device, *Re-*


mote Attestation (RA) has been proposed to detect unexpected modifications in the configuration of loaded binaries and check software integrity. However, typical RA schemes assume the existence of a single Prover and a single Verifier, introducing efficiency and scalability issues for large-scale systems comprising multiple Edge and IoT devices. In this context, swarm attestation [Ambrosin et al., 2020] has been proposed to enable the root Verifier (\mathcal{V}) to check the sanity of a set of swarm devices simultaneously.


Several swarm attestation schemes have been proposed in the literature [Asokan et al., 2015, Ambrosin et al., 2016, Dushku et al., 2023, Le-Papin et al., 2023] to verify the integrity and establish trust among devices in large-scale systems. In such environments, trust is pivotal in the decision-making process as it helps the devices to select collaborators, to assess the trustworthiness of incoming data, and to mitigate threats from untrustworthy or hostile entities. How-


^a <https://orcid.org/0000-0002-2827-6493>


^b <https://orcid.org/0000-0003-2344-044X>


^c <https://orcid.org/0009-0003-2310-921X>


^d <https://orcid.org/0000-0002-4974-9739>

^e <https://orcid.org/0009-0005-2193-4464>

^f <https://orcid.org/0000-0002-2680-6374>

^g <https://orcid.org/0000-0003-2680-4907>

^h <https://orcid.org/0000-0002-4460-9331>

ⁱ <https://orcid.org/0000-0003-0663-2263>

ever, in state-of-the-art solutions, the trust is typically confined to system integrity and dependability aspects, excluding other important dimensions such as identity privacy (anonymity) and evidence privacy.

In such complex and dynamic systems, we often have to make trust decisions based on incomplete, conflicting, or uncertain information. Traditional probabilistic models assume that we have complete knowledge of all possibilities and their respective probabilities, but in dynamic, distributed environments, this is rarely the case. For example, a system might receive conflicting data from different data sources or communication channels. Also, not all data sources are equally reliable, and some may be compromised or faulty.

In such environments, the use of an evidence-based theory for trust assessment becomes essential, because it allows us to explicitly represent uncertainty. Rather than forcing a decision based on incomplete data, we can account for the fact that we do not have enough information, thus reflecting uncertainty in our trust assessments. A core enabler in this direction is the use of robust (swarm) attestation mechanisms for extracting evidence on the status of the target device in a verifiable manner. Depending on the property of the device to be assessed, different types of evidence may be required. In the context of security-related trust properties, trust sources include detection mechanisms, e.g., Intrusion Detection Systems, Misbehaviour Detection Systems, or mechanisms to evaluate the existence and/or enforcement of security claims such as secure boot, configuration integrity etc. In its simplest form, the received evidence can be distinguished into one main category: Binary evidence that allows the verifier to compute whether a security claim is valid or not. For instance, this is the case of the secure boot certificate chain that allows a verifier to make an informed decision that the prover has executed its secure boot process. On the other side, there is evidence that results in claims expressed in the form of a range. This is the case of some Misbehaviour Detection systems that provide a score- e.g., confidence level- about abnormal behaviour of a device. This diversity of the evidence outcome introduces the need for robust attestation mechanisms to enable the quantification of trustworthiness in an efficient and reliable manner.

Typical swarm attestation schemes rely on a *spanning tree* [Asokan et al., 2015, Carpent et al., 2017] to aggregate the attestation results, where swarm devices are attested by neighboring devices based on a parent-child relationship in a tree format with a *centralized* Verifier. To address this fixed parent-child relationship, other works leverage the *broadcasting* ag-

gregate pattern, in which every device broadcasts its response to the attestation challenge to its neighbors [Kohnhäuser et al., 2018]. However, there has been very limited focus on privacy issues of the swarm devices [Abera et al.,]. While some schemes do not provide information about the individual devices or devices causing a failed attestation, they do not explicitly focus on privacy (e.g., [Asokan et al., 2015], [Ambrosin et al., 2016]). Conversely, other attestation schemes only provide information about the devices causing a failed attestation, such as [Ammar et al., 2020]. Moreover, two schemes are proposed in [Carpent et al., 2017], one of which keeps a list of the failed devices, and the other does not.

Swarm attestation faces significant challenges in maintaining identity and evidence privacy, especially in large-scale systems with numerous IoT and edge devices. In this context, we have to be able to make trust decisions based on fresh trustworthiness evidence without impeding the privacy of the overall system. Therefore, it is crucial to conceal each device's identity and the nature of the evidence they present. However, the dynamic nature of such (multi-agent) systems requires that trust assessment is not only based on reasoning under uncertainty (evidence may be incomplete, conflicting, or contain uncertain information on the integrity of the device), but it also ensures that the evidence used to compute trust opinions is verifiable. Trust cannot be randomly assigned or assumed from previous interactions but must be continuously verified and updated through collectable evidence. This principle aligns with the Zero-Trust paradigm, which dictates that no entity should be inherently trusted without validation. Verification of evidence is related to the use of *Trust Anchors* since they are the starting point for verifying the chain of trust.

Consider, for instance, the case of Connected and Cooperative Automated Mobility (CCAM) services in the automotive domain, where it is critical to assess trustworthiness in terms of integrity without compromising the safety-critical profile and privacy of the running services (e.g., Collision Avoidance). Especially when considering the need to share such trusting opinions between different administrative domains. Swarm attestation must effectively verify the status of these devices within the in-vehicle network to ensure that safety decisions are made based on reliable sensor inputs. On the other hand, Verifiers should not ascertain which specific device attests to which property or the exact expected measurement values. Disclosing such information could lead to implementation disclosure attacks, vehicle fingerprinting, or driver behavior tracking. These concepts, while interrelated, require clearer terminological distinctions

to facilitate practical trust assessment and management within decentralized environments of mixed-criticality.

One efficient technique that allows for checking the integrity of the devices while preserving their anonymity is called Direct Anonymous Attestation (DAA). In this work, we extend the DAA to offer two security properties besides anonymity: evidence privacy and traceability. The evidence privacy means that the signer doesn't disclose the signed message, such as the current device's configuration, to the Verifier while still allowing public verifiability. We obtain this by binding the use of the DAA signing key to a restricted policy that only allows the key to sign a message if the policy is satisfied. This subsequently allows the Verifier to trust the attestation results without accessing sensitive information. Traceability enables the correct tracing of a device in case of any failed attestation. The dual protection of device identity and evidence ensures privacy and robust accountability, making it essential for trustworthiness evidence provision in today's evidence-based trust assessment frameworks.

To address the aforementioned challenges, this paper introduces a secure and efficient attestation scheme that can correctly make valid statements about the integrity of both single devices and a swarm of devices in a privacy-preserving but accountable manner. This means that the designed scheme should be able to provide verifiable evidence on the correctness of a swarm by concealing the identities of the devices, and only in the case of a possible compromise detection should it allow for tracing back a failed attestation to the swarm device that caused the failure, instead of isolating the entire swarm.

Contributions. In this paper, we propose the PRIVÉ swarm attestation scheme for privacy-preserving swarm attestation. We leverage trusted computing technologies and strong privacy-enhancing cryptographic protocols (e.g., DAA [Brickell et al., 2004, Chen et al., 2023]), which provide decentralized privacy-preserving attestation using blind group signatures. PRIVÉ considers an interconnected network of heterogeneous IoT and Edge devices to convince a Verifier (\mathcal{V}) of the integrity of the attestation result based on signed attestation evidence of the enrolled swarm devices. Additionally, in its operation, PRIVÉ safeguards the devices' *identity privacy* but also provides *traceability* in case the attested device is deemed compromised. Thus, we do not assume the trustworthiness of the IoT/Edge devices or the Verifier. To the best of our knowledge, we provide the first detailed mathematical analysis for swarm attestation schemes in the *Universally*

Composable (UC) model. This analysis provides proof regarding all envisioned security and privacy requirements in PRIVÉ. Furthermore, we provide a benchmarking analysis to demonstrate the protocol's performance and real-world applicability.

2 PRELIMINARIES

Notation. We present the notations that will be followed throughout the paper in Table 1.

PRIVÉ Building Blocks. To enable a privacy-preserving and accountable swarm attestation approach, PRIVÉ adopts DAA to compute the device attestation with zero knowledge and relies on bilinear aggregation signatures to efficiently aggregate the devices' signatures across the swarm.

Direct Anonymous Attestation (DAA). DAA [Brickell et al., 2004, Brickell et al., 2008], is a platform authentication mechanism that allows privacy-preserving remote attestation of a device associated with a Trusted Component (TC) but does not support the property of traceability. In general, DAA requires an Issuer and a set of Signers, and a set of Verifiers. It includes five algorithms: *Setup*, *Join*, *Sign*, *Verify*, and *Link*. The issuer produces a DAA membership credential for each signer, corresponding to a signature of the signer's identity. A DAA signer consists of the Host and TC pair.

Bilinear Aggregation Signatures. The purpose of designing aggregate signatures is to reduce the length of digital signatures in applications that use multiple signatures [Boneh et al., 2003b]. In a *general aggregate signatures* scheme, the aggregation can be done by anyone without the signers' cooperation. In particular, each given user i signs her message m_i to obtain a signature σ_i . Consider a set of k signatures $\sigma_1, \dots, \sigma_k$ on messages m_1, \dots, m_k under public keys PK_1, \dots, PK_k , respectively. In this case, anyone can use a public aggregation algorithm to compress all k signatures into a single signature σ , whose length is the same as a signature on a single message. To verify it, all the original messages and public keys are needed. In particular, given an aggregate signature σ , public keys PK_1, \dots, PK_k and messages m_1, \dots, m_k , the aggregate verification algorithm verifies whether the aggregate signature σ is valid.

The *bilinear aggregate signature* scheme [Boneh et al., 2003b] enables efficient aggregation, allowing an arbitrary aggregating party unrelated to, and untrusted by, the original signers to combine k pre-existing signatures into a single aggregated signature. The scheme consists of the following algorithms:

Key Generation. For a particular user, pick random $x \leftarrow \mathbb{Z}_p$ and compute $w \leftarrow g_2^x$. $x \in \mathbb{Z}_p$ and $w \in G_2$

Table 1: Notation Summary

Symbol	Description	Symbol	Description
TC	Trusted Component	(x, y)	The Privacy CA private key
E_j	The j^{th} edge device	(X, Y)	The Privacy CA public key
\mathcal{M}_j	TC that corresponds to the j^{th} edge device	(x_L, y_L)	The IoT device long-term key-pair
H_j	The host that corresponds to j^{th} edge device	(x_p, y_p)	The IoT device short-term key-pair (pseudonym)
\mathcal{V}	Verifier	σ_{DAA}^E	DAA signature on the long-term public key y_L
v	Number of edge devices in the swarm	σ_i	i^{th} IoT device signature on its current state m_i
t	TC 's private key	SPK	Signature based Proof of Knowledge
Q_j	E_j 's public key known by the Privacy CA	ch, f	Attestation challenges
(a, b, c, d)	E_j 's credential created by the Privacy CA	ρ, r, s	Random numbers in \mathbb{Z}_q
T_j	E_j 's public tracing key known by the Opener	k	Number of IoT devices in the swarm
T	The Opener's Tracing Key	$\sigma_{[1-n]}$	An aggregated signature of n IoT devices
q	A prime number defining the order of cyclic groups	bsn	A random input in $\{0, 1\}^*$
g_1, g_2, g_3	Generators of the groups G_1, G_2, G_3	\mathbb{M}_i^*	i^{th} IoT devices' golden configuration (legitimate state)
Function	Description	Proofs	Description
H	Hash function: $H : \{0, 1\}^* \rightarrow G_1$	π_{ipk}	Proof of CA public key construction
e	Pairing function: $e : G_1 \times G_2 \rightarrow G_3$	π^1, π^2	Proofs of construction of Q_j and T_j
	$e(g_1, g_2) \rightarrow g_3$	π_j^{CA}	Proof of E_j 's credential construction
		π_L, π_p	Proofs of construction of y_L and y_p , respectively

are the user's private and public keys, respectively.

Signing. For a particular user, given the public key w , the private key x , and a message $m \in \{0, 1\}^*$, compute $h \leftarrow H(w, m)$, where $h \in G_1$, and $\sigma \leftarrow h^x$.

Verification. Given a user's public key w , a message m , and a signature σ , compute $h \leftarrow H(w, m)$. The signature is accepted as valid if $e(\sigma, g_2) = e(h, w)$.

Aggregation. Arbitrarily assign an index i to each user whose signature will be aggregated, ranging from 1 to k . Each user i provides a signature $\sigma_i \in G_1$ on a message $m_i \in \{0, 1\}^*$ of her choice. Compute signature $\sigma \leftarrow \prod_{i=1}^k \sigma_i$.

Aggregate Verification. Given an aggregate signature $\sigma \in G_1$ for a set of users (indexed as before), the original messages $m_i \in \{0, 1\}^*$, and public keys $w_i \in G_2$. To verify the aggregate signature σ , compute $h_i \leftarrow H(w_i, m_i)$ for $1 \leq i \leq k$, and accept if $e(\sigma, g_2) = \prod_{i=1}^k e(h_i, w_i)$ holds.

3 SYSTEM MODEL

We consider a static interconnected network of heterogeneous devices consisting of v edge devices and k IoT devices, following a hierarchical fog computing structure [ope, 2018], as depicted in Fig. 1.

IoT devices (D): An untrusted and resource-constrained device (e.g., a sensor), that is uniquely identified as D_i for $i \in [1, k]$. It is assumed that D has a minimal Trusted Computing Base (TCB) and is connected to only one edge device. The TCB includes (1) a Read-Only Memory (ROM) containing the attestation protocol code, and (2) Secure Key Storage that is read-accessed only by the attestation protocol.

Edge devices (E): An untrusted, powerful device with sizeable computational power and storage ca-

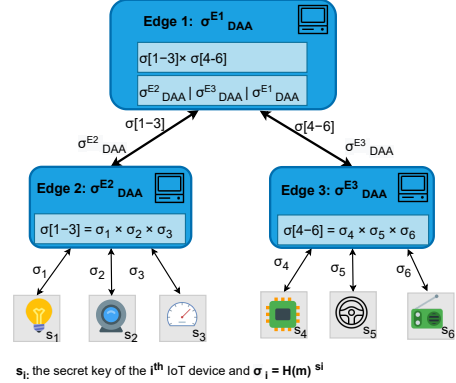


Figure 1: Overview of the Swarm Topology

capacity. It is a combination of a Host H_j (i.e., “normal world”) and a Trusted Component (TC) (i.e., “secure world”) such as Trusted Platform Module (TPM 2.0) [tpm, 2016]. Each edge device is uniquely identified as E_j for $j \in [1, v]$. Each E is a parent of a set of heterogeneous IoT devices and knows its children's legitimate state (i.e., a hash of the binaries). Thus, each edge device authenticates its children's IoT devices.

Verifier (\mathcal{V}): Any third party that initiates the attestation and validates the trustworthiness of the swarm. The Verifier knows the number of IoT devices involved in the swarm. We assume an honest-but-curious Verifier that legitimately performs the verification but will attempt to learn all possible information from the received attestation results.

Privacy Certification Authority (Privacy CA): A trusted third party that acts both as an *Issuer* and *Network Operator*. It is responsible for conducting the secure setup of the fog architecture, verifying the correct creation of all cryptographic primitives, and au-

thorizing the devices to join the network.

Opener (Opener or Tracer): A trusted entity that traces the compromised device(s) when the swarm attestation fails to perform recovery actions.

Threat model. In the context of the system described above, we consider remote software adversaries (Adv_{SW}) that aim to compromise IoT and edge devices in the swarm by exploiting software vulnerabilities and injecting malicious code. Moreover, we consider honest but curious adversaries Adv_{HBC} who are legitimate participants in the swarm but aim to disclose the device's privacy. Additionally, swarms are susceptible to network adversaries (Adv_{NET}) that can forge, drop, delay, and eavesdrop on the messages exchanged among two devices in the swarm. We consider a classic Dolev-Yao (DY) [Dolev and Yao, 1983] adversary with full control over the communication channel between an IoT device and edge device in the swarm, two edge devices, or one edge device and the Verifier. However, in line with other works [Camenisch et al., 2017, Wesemeyer and all, 2020], we assume a perfectly secure channel between the Host and the TC in an edge device. Thus, the Adv_{NET} cannot intercept the interaction between the Host and the TC or use the TC as an oracle. Physical Adversaries (Adv_{PHY}) are beyond our scope.

4 SECURITY REQUIREMENTS

To prove the security properties of PRIVÉ, we employ the UC model introduced in (§ 6.1), for which we define the following high-level security properties:

Anonymity (SP1): It should be possible to conceal the identity of honest Edge and IoT devices from Verifier \mathcal{V} by creating anonymous signatures that perfectly hide the identities of the devices in a swarm.

Traceability (SP2): It should be possible for the Opener to trace the source of an attestation failure back to its source so that a potentially compromised edge device can be revoked. On the other hand, an honest parent edge device can trace its children IoT signatures back to their source IoT devices (SP3).

Correctness (SP4): Honestly generated signatures should always be considered valid, and honest users should not be revoked.

Non-frameability (SP5): It should not be possible for an adversary to create signatures that successfully impersonate an honest device or link to honest signatures. This extends to **Unforgeability (SP6)**, where it should be computationally infeasible to forge a signature σ on a message μ that is accepted by the verification algorithm when no honest device signs μ .

Linkability (SP7): It should be possible for a Verifier to check if any two DAA signatures signed under

the same basename originate from the same edge device without breaching its privacy.

5 THE PRIVÉ PROTOCOL

We present an overview of the main phases of our proposed protocol PRIVÉ without first including the details of internal cryptographic primitives, which will be presented in § 5.1. The setup phase is a one-time offline procedure performed by the Privacy Certification Authority (Privacy CA) to guarantee the secure deployment and enrollment of the devices in the swarm and the creation of the IoT device short-term keys (pseudonyms). The attestation process is then initiated by the Verifier \mathcal{V} that sends a challenge f to one or more edge devices, which will then distribute it recursively to all its (children) swarm devices. Then, each IoT Device D_i produces and signs its attestation evidence (attestation claim) using a pre-certified pseudonym. All these claims are sent to the parent Edge Device, E_j , that aggregates them, appends its DAA signature, and sends the aggregated claims together with its DAA signature to the root edge (such as Edge 1 in Figure 1). When the root edge receives all other edges' contributions, it creates its DAA signature, aggregates the received IoT aggregated claims and appends all the DAA signatures, including its own. This constitutes the swarm signature. The swarm signature is then forwarded to \mathcal{V} . After obtaining the swarm attestation report, \mathcal{V} verifies DAA signatures and checks the total number of IoT signatures to ensure that all IoT devices have been included in the report. In the case of a non-valid DAA signature or a missing IoT signature, \mathcal{V} leverages the novel traceability feature of PRIVÉ to trace the result back to the compromised device.

5.1 Detailed Description of PRIVÉ

In this section, we provide a detailed description of PRIVÉ. The required notation is presented in Tab. 1. The protocol consists of the **Setup and Join** (§ 5.1.1), **Attestation** (§ 5.1.2), and **Verification** (§ 5.1.3) phases.

5.1.1 Setup and Join Phases

The setup phase starts with the Privacy CA being equipped with the necessary key pairs $x, y \leftarrow \mathbb{Z}_q$, where $X = g_2^x$ and $Y = g_2^y$. The Privacy CA, if needed, can prove the well-formed construction of the secret key (x, y) through the creation of a proof of knowledge π_{ipk} (that can also be signed by the Root Certification Authority in such systems). (X, Y) are public

parameters shared by the Privacy CA and other system parameters.

Edge-Join. The edge device Enrollment process starts from an edge device E_j , where $j \in [1, v]$, that sends an enrollment request to the Privacy CA for joining the target service graph chain. The Privacy CA chooses a fresh nonce p and sends it to E_j , who forwards it to its embedded TC denoted by \mathcal{M}_j . Then, \mathcal{M}_j chooses a secret key $t \leftarrow \mathbb{Z}_q$, sets its public key $Q_j = g_1^t$. We use a policy that can be satisfied if a selection of the Platform Configuration Registers (PCR)s matches a predetermined value, referencing a trusted state. We use PolicyPCR commands (of the underlying Root-of-Trust) to ensure that the edge device signing key is inoperable if the integrity of the Edge device is compromised (cf. Section 8 for further details on the implementation of such commands in the case of a Trusted Platform Module (TPM) as the host secure element). \mathcal{M}_j can also compute π_j^1 as a proof of construction of Q_j . This proof, Q_j , can be signed with the embedded TC's endorsement key, thus elevating it to a verifiable credential over the correct construction and binding of the secret key t . The pair (Q_j, π_j^1) (alongside the public part of the TC's root certificate) is then sent to the Privacy CA who verifies π_j^1 and checks whether the edge device is eligible to join, i.e. the edge device DAA key has not been registered before. Upon validation, the Privacy CA picks a random $r \leftarrow \mathbb{Z}_q$ and generates E_j 's DAA credential (a, b, c, d) by setting $a = g_1^r, b = a^r, c = a^x Q_j^{xy}$ and $d = Q_j^{ry}$. The Privacy CA also generates a proof π_j^{CA} on the credential construction.

Generating the Tracing Keys by the Opener. Each \mathcal{M}_j sets $T_j = g_2^t$ and computes π_j^2 as a proof of construction of T_j . \mathcal{M}_j sends $(T_j, Q_j, \pi_j^1, \pi_j^2)$ via E_j to the Opener. The Opener verifies π_j^1 and π_j^2 and makes sure that T_j and Q_j link to the same TC by checking if the following holds:

$$e(Q_j, g_2) = e(g_1, T_j) \quad (1)$$

To convince the Opener that T_j is a correct key, \mathcal{M}_j may also send to the Opener the DAA credential of Q_j issued by the Privacy CA. After receiving all edge devices T_j , for all $j \in [1, v]$, and their corresponding public keys Q_j , the Opener sets its tracing key T as follows: $T = \{T_1, T_2, \dots, T_v\}$ and keeps the (Q_j, T_j) pairs for all $j \in [1, v]$.

IoT-Join. Each IoT device is equipped with a long-term key pair $(x_L \in \mathbb{Z}_q, y_L)$ whose public part $y_L = g_1^{x_L}$ represents the IoT device's identity and is certified by the Privacy CA. Let π_L represent a proof of construction of $y_L = g_1^{x_L}$. The enrollment of IoT devices consists of the following steps:

1. Each edge device certifies its children's IoT long-term key by creating a DAA signature σ_{DAA}^L on y_L .
2. Once the long-term key is certified, each IoT device creates short-term random keys $(x_p \in \mathbb{Z}_q, y_p = g_2^{x_p}, \pi_p)$, for each integer $p \in [1, \dots, P]$ where π_p is a proof of construction of y_p and P is the total number of pseudonyms (short-term random keys) that will be certified by the edge device. This is called pseudonymity, which is the ability of an edge device to use a resource or service without disclosing the IoT user's identity while still being accountable for that action.
3. The IoT device self-certifies its public key y_p by creating a signature σ_L^p on y_p using its own long-term secret key x_L . This signature prevents any other device (including edge devices and authorities) from signing on behalf of the IoT device without knowing the IoT device's long-term secret signing key x_L . Finally, the IoT device sends $(y_L, y_p, \sigma_{DAA}^L, \pi_p)$ for each $p \in [1, \dots, P]$ to its mother.
4. The edge device verifies π_p and σ_L^p on y_p for each $p \in [1, \dots, P]$. Upon successful verification, the edge device keeps $(y_p, y_L, \sigma_{DAA}^L, \sigma_L^p)$ in its records to be able to link each set of certified IoT pseudonyms to their long-term key for each child IoT device. This is particularly useful in cases where an edge device needs to trace the identity of its children's IoT signatures.

We remove the index p from the remaining protocol description and use σ_i and (x_i, y_i) to denote the D_i 's signature and short-term key, respectively.

5.1.2 Attestation Phase

The swarm attestation is initiated by a Verifier \mathcal{V} that sends a challenge f to an edge device, which will then distribute it recursively to all devices in the swarm. The IoT devices sign their attestation results and report them to their parent edge devices.

IoT Device Signature. When an IoT device receives an attestation challenge f , it concatenates its current configuration m_i^* with the challenge f . Then, signs the attestation response $m_i = (m_i^* || f)$ using one of its short-term keys x_i previously certified by its parent edge device during the enrollment procedure. Each x_i is only used once to achieve IoT identity privacy. The IoT device computes a BLS signature as follows:

$$\sigma_i = H(m_i)^{x_i} \quad (2)$$

The IoT device sends σ_i and y_i to its parent edge. The parent edge device checks that every received y_i is certified, relying on its records. The edge then aggregates all the c received children signatures $\sigma_1, \dots, \sigma_c$ in one signature $\sigma_{[1-c]}$ as follows $\sigma_{[1-c]} = \sigma_1 \times \dots \times \sigma_c =$

$H(m_1)^{x_1} \times \dots \times H(m_c)^{x_c}$. When the attestation of a given IoT device is missing, the edge device will not include it in the aggregation.

Edge Device Signature & Traceability. The creation of a traceable DAA signature starts with an edge device E_j , with an embedded trusted component \mathcal{M}_j , that gets a Verifier basenane bsn defined as a random string $\{0, 1\}^*$ to enable traceability and linkability of DAA signatures. The signature generation follows the following steps:

1. To sign a message μ with respect to the basenane bsn , the edge device randomizes its credential by choosing a random $s \leftarrow \mathbb{Z}_q$, sets $(a', b', c', d') \leftarrow (a^s, b^s, c^s, d^s)$, and then sends (μ, bsn, b') to \mathcal{M}_j .
2. Next, \mathcal{M}_j calculates the link token $nym = (bsn)^t$. E_j and \mathcal{M}_j calculate a Signature Based Proof of Knowledge SPK of the secret key t and its credential as follows:

$$SPK\{t : nym = H(bsn)^t, d' = b''\}(bsn, \mu) \quad (3)$$

\mathcal{M}_j samples $\omega \leftarrow \mathbb{Z}_q$, sets $K = H(bsn)^\omega$, $J = b'^\omega$ and sends (nym, K, J) to E_j that calculates the challenge $ch = H(a', b', c', d', nym, K, J, \mu)$ in \mathbb{Z}_q and sends it to \mathcal{M}_j . \mathcal{M}_j then creates a Schnorr signature $\delta = \omega + ch \cdot t$. Finally, the SPK has a form of (δ, ch) and is included in the DAA signature.

3. The DAA signature is $(a', b', c', d', SPK, nym)$. To this end, the swarm signature is $(\sigma_{[1-k]}, \sigma_{DAA}^j, y_i) \forall i \in [1-k]$ and $j \in [1-v]$, where k and v are the total numbers of IoT and edge devices in the swarm, respectively and $\sigma_{[1-k]}$ denotes the aggregation of all the IoT signatures in the swarm. The root edge performs this aggregation. If the attestation of some IoT devices is unsuccessful or missing, the mother edge device does not aggregate such attestations. The final swarm aggregated signature will be as follows: $(\sigma_{[1-l]}, \sigma_{DAA}^j, y_i), \forall i \in [1-l]$ with $l < k$.

5.1.3 Verification Phase

Attestation Report Verification. After receiving an attestation report, the Verifier checks the DAA signature of each edge device $\sigma = (a', b', c', d', nym, SPK = (\delta, ch))$ on a message μ w.r.t. a basenane bsn that is published with the signature. In particular, the Verifier verifies SPK w.r.t. (μ, bsn) and nym , and checks that $a' \neq 1$, $e(a', Y) = e(b', g_2)$, $e(c', g_2) = e(a'd', X)$.

We assume that the verifiers have access to the i^{th} IoT devices' golden configuration, denoted by \mathbb{M}_i^* , $\forall i \in [1, k]$. The Verifier calculates $H(\mathbb{M}_i)$, with $\mathbb{M}_i =$

$(\mathbb{M}_i^* | f)$, and checks the following equation:

$$e(\sigma_{[1-k]}, g_2) \stackrel{?}{=} e\left(H(\mathbb{M}_1), y_1\right) \times \dots \times e\left(H(\mathbb{M}_k), y_k\right) \quad (4)$$

that is only valid if the IoT device's current configuration m_i , used in Equation 2, matches the golden one \mathbb{M}_i without knowing the value of m_i ; this offers IoT evidence privacy. The Verifier claims that the swarm attestation has failed if the aggregate result has less than k aggregated signatures or if the verification of Equation 4 fails. The verifier also checks that $ch = H(a', b', c', d', nym, H(bsn)^\delta, nym^{-ch}, b'^\delta d'^{-ch}, \mu)$.

Further, the Verifier can interact with the Opener to initiate the tracing of the device with a failed attestation.

Link. When a Verifier can access different swarm attestation reports, she can check whether any two DAA signatures originate from the same edge device. Specifically, the Verifier checks if both signatures are signed under the same basenane known by the Verifier. More formally, given $\sigma_1 = (a'_1, b'_1, c'_1, d'_1, nym_1, SPK_1)$ and $\sigma_2 = (a'_2, b'_2, c'_2, d'_2, nym_2, SPK_2)$ on a message μ w.r.t. a basenane bsn . In this setting, the output is 1 when both signatures are valid and $nym_1 = nym_2$; otherwise, the output is 0.

The linkability of the IoT devices in a swarm attestation relies on the static topology of the swarm; each parent edge device has in its records the set of all certified pseudonyms associated with the corresponding IoT device long-term key (used as the identity of the IoT device) for each child IoT device. Thus, whenever an IoT device creates two signatures in two different SA instantiations, the edge device can identify whether these two signatures originate from the same IoT device (the edge device uses traceability to support linkability of IoT devices).

Tracing/Opening. In PRIVÉ, traceability consists of two levels: The first level of swarm attestation traceability relies on the traceability of the DAA signature, which can be done by the Opener using its tracing key. We achieve this by adding a traceability requirement for the existing DAA scheme [Camenisch et al., 2016] to obtain a novel *Traceable DAA* protocol that meets our security and privacy requirements. The second level relies on the static swarm topology that links each parent to their children. Tracing the IoT device can be done by the edge device with the knowledge of the IoT long-term key corresponding to the certified short-term pseudonyms. Specifically, starting with a DAA signature σ_{DAA} with a corresponding link token $nym = H(bsn)^t$, the Opener uses its tracing key $T = \{T_1, T_2, \dots, T_v\}$ to check the following equation:

$$e(nym, g_2) = e(H(bsn), T_j) \quad \forall j \in [1-v] \quad (5)$$

A successful verification allows the Opener to recover \mathcal{M}_j 's public key Q_j corresponding to T_j in its records, as explained previously during the setup of the Tracing Keys by the Opener. Once the edge device's identity is recovered, it is easy for the edge device to recover any of its children's long-term public keys that are certified by this edge device.

Revocation. Any Revocation Authority (RA), which may also be the Opener, can remove misbehaving edge devices without revealing the edge device's identity. The intuition behind the revocation scheme is to be able to deactivate any DAA credential of an edge device created during its enrollment [Larsen et al., 2021]. We assume that a Revocation Authority RA has access to a set of revoked DAA keys KRL. The revocation authority checks that the DAA key used to create the DAA signature is not in the revocation list. This is done by checking that: $\forall t^* \in \text{KRL}, \text{nym} \neq H(\text{bsn})^{t^*}$. PRIVÉ also supports revocation of the IoT devices by their parent edge device. We assume the edge device has access to a set of IoT revoked keys called the IoT-Key Revocation List IoT-KRL. It can then check that each of its children's IoT signatures were not produced by any key in IoT-KRL. Relying on the trust of the edge devices and the accuracy of the updated IoT-KRL, edge devices would also be able to correctly revoke the IoT devices that their keys are IoT-KRL.

6 PRIVÉ Security Analysis

In this section, we first introduce the Universally Composable (UC) framework (§ 6.1), then we model PRIVÉ in the Universally Composable (UC) framework (§ 6.2), capturing all the security and privacy requirements defined in § 4.

6.1 Methodology

Due to the complexity of the PRIVÉ scheme, it isn't easy to perform a complete security proof based on traditional formal verification and symbolic modeling techniques as the specified models might prove challenging to analyze and verify even in computational rich machines [Meier et al., 2013]. Therefore, we employ the Universal Composability (UC) model, based on which we aim to equate the real-world network topology to an ideal-world model, both being indistinguishable from each other from the perspective of an adversary, in which it is possible to break down the proposed scheme into simpler building blocks with provable security. Specifically, the security definition of our swarm attestation protocol is given with respect to an ideal functionality \mathcal{F} . This is depicted

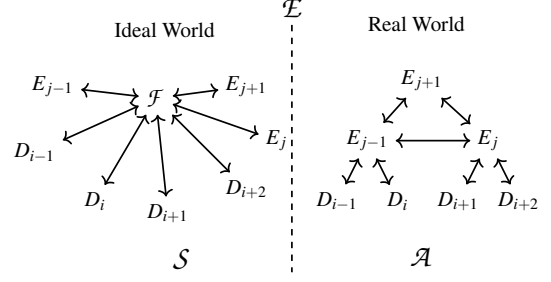


Figure 2: UC security model: the Real and the Ideal world executions are indistinguishable to the environment \mathcal{E} .

in Figure 2 for a swarm of three edges and four IoT devices. However, the UC model can be generalized to any network topology and swarm size. In UC, an environment \mathcal{E} should not be able to distinguish, with a non-negligible probability, between the two worlds:

The *real world*, where each of the PRIVÉ parties: an Issuer I , a parent edge E_j , a child IoT device D_i , and a Verifier \mathcal{V} executes its assigned part of the real-world protocol PRIVÉ denoted by Π . The network is controlled by an adversary \mathcal{A} that communicates with the environment \mathcal{E} . The *ideal world*, in which all parties forward their inputs to a trusted entity, called the ideal functionality \mathcal{F} , which internally performs (in a trustworthy manner) all the required tasks and creates the parties' outputs in the presence of a simulator \mathcal{S} .

The endmost goal is to ensure that PRIVÉ *securely realizes* all internal cryptographic tasks if for any real-world adversary \mathcal{A} that interacts with the swarm, running Π , there exists an ideal world simulator \mathcal{S} that interacts with the ideal functionality \mathcal{F} so that no probabilistic polynomial time environment \mathcal{E} can distinguish whether it is interacting with the real \mathcal{A} or the ideal world adversary \mathcal{S} . Another key point of UC towards reducing the computational complexity of the specified protocol is the composition theorem that preserves the security of PRIVÉ, even if it is arbitrarily composed with other instances of the same or different protocols.

6.2 PRIVÉ UC Model

We now explain our security model in the UC framework based on the ideal functionality \mathcal{F} originally proposed in [Camenisch et al., 2016] and later adopted in [Chen et al., 2019, El Kassem et al., 2019]. We assume that \mathcal{F} internally runs the sub-functionalities defined in [Camenisch et al., 2016]. In protocol's join phase, we adopt the key binding protocol introduced in [Chen et al., 2024], which ensures an authenticated channel between the TC and the Issuer even in the presence of a corrupt Host; therefore, in contrast to [Camenisch et al., 2016], we don't need

to model the semi-authenticated channel.

The UC framework lets us focus the analysis on a single protocol instance with a globally unique session identifier sid . \mathcal{F} uses session identifiers of the form $sid = (I, sid')$ for some Issuer I and a unique string sid' . In the real world, these strings are mapped to the Issuer's public key, and all parties use sid to link their stored key material to the particular Issuer. We define the Edge-JOIN, the IoT-JOIN, and the SIGN sub-session identifiers $jsid_j$, $jsid_i$ and $ssid$, respectively, to distinguish several join and sign sessions that might run in parallel. We define two "macros" to determine whether a secret key key is consistent with the internal functionality records. This is checked at several places in the ideal functionality interfaces and depends on whether key belongs to an honest or corrupt party. The first macro CheckkeyHonest is used when the functionality stores a new key that belongs to an honest party and checks that none of the existing valid signatures are identified as belonging to this party. The second macro CheckkeyCorrupt is used when storing a new key that belongs to a corrupt party and checks that the new key does not break the identifiability of signatures, i.e., it checks that there is no other known key^* , unequal to key , such that both keys are identified as the owner of a signature. Both functions output a bit b , where $b = 1$ indicates that the new key is consistent with the stored information, whereas $b = 0$ signals an invalid key. We assume that the Host H_j represents the same entity as E_j . Thus, \mathcal{M}_j and E_j correspond to the TC and the Host in an edge device E_j . Finally, our model defines lists that will be kept in the ideal functionality record for the consistency of the joining devices, keys, signatures, verification results, and key revocation lists namely: EdgeMembers, EdgeKeys, EdgeSigned, EdgeVer, IoTMembers, IoTKeys, IoTSigned, IoTVer KRL and IoTKRL. Next, we present the interfaces of PRIVÉ ideal functionality \mathcal{F} .

SETUP: On input (SETUP, sid) from an Issuer I , output (SETUP, sid) to \mathcal{S} . \mathcal{F} then expects the adversary \mathcal{S} to provide the following algorithms, namely (*Kgen*, *Sign*, *Verify*, *Link*, and *Identify*), that will be used inside the functionality as follows:

Kgen: It consists of two probabilistic sub-algorithms: *Kgen* that generates keys tsk and dsk for honest TCs and IoT devices, respectively. Note that E_j and D_i are uniquely identified by their keys tsk_j and dsk_i , respectively. We remove the indices from the keys in our security model description. *PSEUDO-Kgen* that takes dsk as an input and generates a random short-term key pair (s, o) for honest IoT devices.

Sign: A probabilistic algorithm used for honest TCs

and IoT devices to create the type of signature required by the corresponding operation. It consists of the following sub-algorithms:

1. $\text{sig}_{\text{DAA}}(tsk, \mu, \text{bsn})$: On input of a secret key tsk , a message μ and a basename bsn , it outputs a DAA signature σ_{DAA}^j on behalf of E_j .
2. $\text{sig}_{\text{IoT}}(dsk, m)$: It creates signatures on behalf of D_i . It runs *PSEUDO-Kgen* to create a short-term key pair (s, o) , then creates a signature σ_i on a message m using the secret short-term key s .

Aggregate: A deterministic algorithm that aggregates the IoT signatures using *agg* algorithm that takes as inputs c IoT signatures $\sigma_1, \sigma_2, \dots, \sigma_c$ and outputs an aggregated signature $\sigma_{[1-c]}$.

Verify: A deterministic algorithm that outputs a binary result on the correctness of a signature. It consists of two sub-algorithms:

1. $\text{ver}_{\text{DAA}}(\sigma_{\text{DAA}}, \mu, \text{bsn})$: On input of a signature σ_{DAA} , a message μ and a basename bsn , it outputs $b = 1$ if the signature is valid, 0 otherwise.
2. $\text{ver}_{\text{IoT}}(\sigma, m, o)$: On input of a signature σ , a message m and a public short-term key o , it outputs $b = 1$ if the signature is valid, 0 otherwise.

Link: $(\sigma_1, \mu_1, \sigma_2, \mu_2)$: A deterministic algorithm that checks if two signatures originate from the same device. In the case of DAA signature linkability checks, an extra parameter bsn is required as an input. It outputs 1 if the same device generates σ_1 and σ_2 , and 0 otherwise.

Identify: A deterministic algorithm that ensures consistency with the ideal functionality \mathcal{F} 's internal records by connecting a signature to the key used to generate it. It consists of two sub-algorithms:

1. $\text{identify}_{\text{DAA}}(tsk, \sigma_{\text{DAA}}, \mu, \text{bsn})$: It outputs 1 if tsk was used to produce σ_{DAA} , 0 otherwise.
2. $\text{identify}_{\text{IoT}}(\sigma, m, dsk, s)$: It outputs 1 if σ was produced by an IoT device D , with an identity key dsk , on a message m using the short term key s , 0 otherwise.

On input (ALGORITHMS, sid) from \mathcal{S} , the ideal functionality checks that *Aggregate*, *Verify*, *Link*, and *Identify* are deterministic, store the algorithms and output (SETUPDONE, sid) to I .

Edge-JOIN: On input (JOIN, sid , $jsid_j$, \mathcal{M}_j) from E_j , create a join session record $\langle jsid_j, \mathcal{M}_j, E_j \rangle$ and output (JOINPROCEED, sid , $jsid_j$, \mathcal{M}_j) to I . On input (JOINCOMPLETE, sid , $jsid_j$, tsk) from \mathcal{S} :

- Abort if I or \mathcal{M}_j is honest and a record $\langle \mathcal{M}_j, *, * \rangle \in \text{EdgeMembers}$ already exists.
- If \mathcal{M}_j and E_j are honest, set $tsk \leftarrow \perp$.
- Else, verify that the provided tsk is eligible by:
- CheckkeyHonest(tsk) = 1 if \mathcal{M}_j is honest and E_j is corrupt, or CheckkeyCorrupt(tsk) = 1 if \mathcal{M}_j is

corrupt and E_j is honest.

Insert $\langle \mathcal{M}_j, E_j, ts_k \rangle$ into EdgeMembers and output (JOINED, $sid, jsid_j$) to E_j .

IoT-JOIN On input (JOIN, $sid, jsid_{ij}, D_i, E_j$) from D_i , create a join session record $\langle jsid_{ij}, D_i, E_j \rangle$ and output (JOINPROCEED, $sid, jsid_{ij}, D_i$) to E_j .

On input (JOINCOMPLETE, $sid, jsid_{ij}, dsk$) from S :

- Abort if I or D_i is honest and a record $\langle D_i, *, * \rangle \in \text{IoTMembers}$ already exists.
- If D_i and E_j are honest, set $dsk \leftarrow \perp$.
- Else, verify that the provided dsk is eligible by $\text{CheckkeyHonest}(dsk) = 1$ if D_i is honest, or $\text{CheckkeyCorrupt}(dsk) = 1$ if D_i is corrupt.

Insert $\langle D_i, E_j, dsk \rangle$ into IoTMembers and output (JOINED, $sid, jsid_{ij}, D_i$) to E_j .

IoT-SIGN For all $i \in [1, c_j]$, where c_j is the number of E_j 's children IoT devices participating in the swarm attestation. On input (SIGN, $sid, ssid, m_i$) from D_i , with m_i is the message to be signed by a child D_i , if I is honest and no entry $\langle D_i, E_j, * \rangle$ in IoTMembers, abort. Else, create a sign session record $\langle ssid, D_i, E_j, m_i \rangle$ and output (SIGNPROCEED, $sid, ssid$) to D_i .

- On input (SIGNCOMPLETE, $sid, ssid, \sigma_i$) from S , if D_i and E_j are honest, ignore the adversary's signature σ_i and internally generate the signature for a fresh dsk or established dsk :
 - Retrieve dsk from IoTKeys, if no key exists, set $dsk \leftarrow \text{Kgen}()$ and check $\text{CheckkeyHonest}(dsk) = 1$.
 - Compute signature $\sigma_i \leftarrow \text{sig}_{\text{IoT}}(dsk, m_i)$ that internally runs PSEUDO-Kgen to create a short-term key pair (s_i, o_i) for D_i , then check that $\text{ver}_{\text{IoT}}(\sigma_i, m_i, o_i)$ and $\text{identify}_{\text{IoT}}(\sigma_i, m_i, dsk, s_i)$ are both equal to 1.
 - Check that there is no $D' \neq D_i$ with key dsk' registered in IoTMembers or IoTKeys with $\text{identify}_{\text{IoT}}(\sigma_i, m_i, dsk', s_i) = 1$.
- Store $\langle D_i, dsk, (s_i, o_i) \rangle$ in IoTKeys.
- Compute $\sigma_{[1-c_j]}^j \leftarrow \text{agg}(\sigma_1, \sigma_2, \dots, \sigma_{c_j})$.

• Output $(sid, ssid, \sigma_{[1-c_j]}^j, \{\sigma_i, o_i\}_{i \in [1, c_j]})$ to E_j .

AGGREGATION Calculate the aggregation of the v (total number of edges in a swarm) aggregated signatures as follows: $\sigma_{[1-v]} \leftarrow \text{agg}(\sigma_{[1-c_1]}^1, \dots, \sigma_{[1-c_v]}^v)$, and store $\langle \sigma_{[1-v]}, \sigma_{[1-c_j]}^j, (\sigma_i, m_i, D_i, o_i)_{i \in [1, l]} \rangle$ in IoTSigned, where $l = \sum_{j=1}^v c_j$.

Edge-SIGN On input (SIGN, $sid, ssid, \mathcal{M}_j, \mu, \text{bsn}$) from E_j , where $j \in [1, v]$. If I is honest and no entry $\langle \mathcal{M}_j, E_j, * \rangle$ exists in EdgeMembers, abort. Else, create a sign session record $\langle ssid, \mathcal{M}_j, E_j, \mu, \text{bsn} \rangle$ and output (SIGNPROCEED, $sid, ssid, \mu, \text{bsn}$) to \mathcal{M}_j .

On input (SIGNCOMPLETE, $ssid, \sigma_{DAA}^j$) from S :

- If \mathcal{M}_j and E_j are honest, ignore the adversary's signature and internally generate the signature for a fresh or established tsk :
 - If $\text{bsn} \neq \perp$, retrieve tsk from $\langle \mathcal{M}_j, \text{bsn}, ts_k \rangle \in \text{EdgeKeys}$. If no such tsk exists or $\text{bsn} = \perp$, set $tsk \leftarrow \text{Kgen}()$. Check $\text{CheckkeyHonest}(tsk) = 1$ and store $\langle \mathcal{M}_j, \text{bsn}, ts_k \rangle$ in EdgeKeys.
 - Compute $\sigma_{DAA}^j \leftarrow \text{sig}_{DAA}(tsk, \mu, \text{bsn})$ and check $\text{ver}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn}) = 1$.
 - Check $\text{identify}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn}, ts_k) = 1$ and check that there is no $\mathcal{M}' \neq \mathcal{M}_j$ with key tsk' registered in EdgeMembers or EdgeKeys with $\text{identify}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn}, ts_k') = 1$.
- If E_j and \mathcal{M}_j are honest, store $\langle \sigma_{DAA}^j, \mu, \text{bsn}, \mathcal{M}_j, E_j \rangle$ in EdgeSigned and output (SIGNATURE, $sid, ssid, \sigma_{DAA}^j$) to E_j .

Edge-VERIFY On input (VERIFY, $sid, \mu, \text{bsn}, \sigma_{DAA}^j, \sigma_{[1-l]}, o_i, \text{KRL}$), $i \in [1, l]$ and $j \in [1, v]$ from \mathcal{V} .

- For each j , retrieve all pairs (tsk, \mathcal{M}_j) from $\langle \mathcal{M}_j, *, ts_k \rangle \in \text{EdgeKeys}$ where $\text{identify}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn}, ts_k) = 1$. Set $f_j = 0$ if at least one of the following conditions holds:
 - CHECK 1. More than one key tsk was found.
 - CHECK 2. I is honest and no pair (tsk, \mathcal{M}_j) was found.
 - CHECK 3. There is an honest \mathcal{M}_j but no entry $\langle *, \mu, \text{bsn}, \mathcal{M}_j \rangle \in \text{EdgeSigned}$ exists.
 - CHECK 4. There is a $tsk^* \in \text{KRL}$ where $\text{identify}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn}, ts_k^*) = 1$ and no pair (tsk, \mathcal{M}_j) for an honest \mathcal{M}_j was found.
- If $f_j \neq 0$, set $f_j = \text{ver}_{DAA}(\sigma_{DAA}^j, \mu, \text{bsn})$.
- Add $\langle \sigma_{DAA}^j, \mu, \text{bsn}, \text{KRL}, f_j \rangle$ to EdgeVer, then proceed to verify the children IoT signatures.

IoT-VERIFY Output 0 if $l \neq k$, else, retrieve $\langle \sigma_{[1-v]}, \sigma_{[1-c_j]}^j, (\sigma_i, m_i, D_i, o_i)_{i \in [1, l]} \rangle$ from IoTSigned. For each i retrieve all pairs (dsk, D_i) from IoTMembers and $\langle D_i, dsk, (s_i, o_i) \rangle \in \text{IoTKeys}$ where $\text{identify}_{\text{IoT}}(\sigma_i, m_i, dsk, (s_i, o_i)) = 1$. Set $f_i = 0$ if at least one of the following conditions holds:

- CHECK1. More than one key dsk was found.
- CHECK2. If the parent edge E_j is honest, no (dsk, D_i) was found.
- CHECK3. There is an honest D_i but no entry $\langle *, *, (\sigma_i, m_i, D_i, *) \rangle \in \text{IoTSigned}$ exists.
- CHECK4. There is a $dsk^* \in \text{IoTKey}$ where $\text{identify}_{\text{IoT}}(\sigma_i, m_i, dsk^*, (s_i, o_i)) = 1$ and no $(dsk, D_i, (s_i, o_i))$ for honest D_i was found.
- If $f_i \neq 0$, set $f_i = \text{ver}_{\text{IoT}}(\sigma_i, m_i, o_i)$ and add $\langle \sigma_i, m_i, \text{IoTKey}, f_i \rangle$ to IoTVer.
- Output (VERIFIED, 0, $sid, \sigma_{DAA}^j, \sigma_{[1-k]}, y_i$), to \mathcal{V}

if there exists at least one f_i or f_j equals to 0 for all $j \in [1, v]$ and $i \in [1, l]$. Otherwise (VERIFIED, 1, $sid, \sigma_{DAA}^j, \sigma_{[1-l]}, o_i$).

Edge-LINK: On input (LINK, $sid, \sigma, \mu, \sigma', \mu', bsn$) from some party \mathcal{V} with $bsn \neq \perp$, else:

- Output \perp to \mathcal{V} if at least one signature tuple (σ, μ, bsn) or (σ', μ', bsn) is not valid.
- For each tsk in EdgeMembers and EdgeKeys compute $b = \text{identify}_{DAA}(\sigma, \mu, bsn, tsk)$ and $b' = \text{identify}_{DAA}(\sigma', \mu', bsn, tsk)$, set $f = 0$ if $b \neq b'$ and $f = 1$ if $b = b' = 1$.
- If f is not defined yet, set $f = \text{link}(\sigma, \mu, \sigma', \mu', bsn)$. Output (LINK, sid, f) to \mathcal{V} .

Edge-TRACE On input (TRACE, $sid, \sigma_{DAA}^j, \mu, bsn$) from \mathcal{V} , retrieve all pairs (tsk, \mathcal{M}_j) from $\langle \mathcal{M}_j, *, tsk \rangle \in \text{EdgeKeys}$ where $\text{identify}_{DAA}(\sigma_{DAA}^j, \mu, bsn, tsk) = 1$. Output (TRACE, sid, E_j) to \mathcal{V} .

IoT-LINK: On input (LINK, $sid, \sigma, m, \sigma', m'$) from E_j , output \perp to E_j if at least one signature tuple (σ, m) or (σ', m') is not valid (verified via the `verify` interface with `IoTKey = 0`), else:

- For each (dsk, s_i) in IoTMembers and IoTKeys compute $b = \text{identify}_{IoT}(\sigma, m, dsk, s)$ and $b' = \text{identify}_{IoT}(\sigma', m', dsk', s')$ and set $f = 0$ if $b \neq b'$ and $f = 1$ if $b = b' = 1$.
- If f is not defined yet, set $f = \text{link}(\sigma, m, \sigma', m')$.
- Output (LINK, sid, f) to E_j .

IoT-TRACE On input (TRACE, sid, σ_i, m_i) from E_j , retrieve all pairs (dsk, D_i) from IoTMembers and $\langle D_i, dsk, (s_i, o_i) \rangle \in \text{IoTKeys}$ where $\text{identify}_{IoT}(\sigma_i, m_i, dsk, (s_i, o_i)) = 1$. Output (TRACE, sid, D_i) to E_j .

6.3 Realizing Security Requirements

The complete security model of the intelligence part of PRIVÉ expressed in UC terms, including the ideal functionality interfaces for all the internal phases and some essential checks, is presented in § 6.2. Before proceeding with the UC security proof in § 7, we first map the modeled interfaces to the security properties outlined in § 4.

Anonymity (SP1): The anonymity of honest edge devices, equipped with a valid TC, and their benign (children) IoT devices is guaranteed by \mathcal{F} due to the random choice of tsk (E_j 's private key) and dsk (D_i 's private key) for the construction of every DAA and IoT signature as part of the **Edge/IoT-SIGN** interfaces. In case of corrupt devices (either Edge or IoT), \mathcal{S} provides the signature, which conveys the signer's identity, as the signing key is extracted from the respective device key pair. This reflects that *the anonymity of the whole swarm is only guaranteed if*

both the Edge and IoT devices are honest.

Traceability (SP2): CheckkeyHonest described in § 6 prevents registering an honest tsk/dsk in the **Edge/IoT-JOIN** interfaces that match an existing signature, so that conflicts can be avoided and signatures can always be traced back to the origin devices. Moreover, in the context of *Revocation* (SP3), CHECK4 guarantees that valid signatures in the **Edge/IoT-VERIFY** interfaces are not revoked due to the *identify* algorithm being deterministic.

Correctness (SP4): When an honest device successfully creates a signature, honest Verifiers will always accept this signature. This is since honestly generated signatures in the **Edge/IoT-SIGN** interfaces pass through `verDAA` and `verIoT` checks before being output to the external environment.

Non-frameability (SP5): The non-frameability property guarantees that this signature cannot be linked to a legitimate one created by the target device. In this context, CHECK3 in the **Edge/IoT-VERIFY** interfaces ensures that if the edge (respectively IoT) device is honest, then no adversary can create signatures that are identified to be signed by the edge (respectively IoT) device. This extends to the **unforgeability (SP6)** property, which dictates that it is computationally infeasible to forge signatures.

Linkability (SP7): CheckkeyCorrupt defined in § 6.2 prevents a corrupt tsk/dsk , matching with an existing signature, from being added to the lists of existing members and keys, thus enabling the correct linkability of signatures.

7 PRIVÉ Security Proof (Sketch)

Since we have defined the UC security model of PRIVÉ, we can now proceed with the security proof of the proposed scheme, leveraging the Discrete Logarithm (DL) and Decisional Diffie-Hellman (DDH) assumptions. This can be expressed as follows:

Definition 7.1. (*The Discrete Logarithm (DL) assumption [McCurley, 1990]*) Given $y \in G_2$, find an integer x such that $g_2^x = y$.

Definition 7.2. (*Decisional Diffie-Hellman (DDH) assumption [Boneh, 1998]*) Let G be a group generated by some g . Given g^a and g^b , for random and independent integers $a, b \in \mathbb{Z}_q$, a computationally bounded adversary cannot distinguish g^{ab} from any random element g^r in the group.

Theorem 7.1. *The PRIVÉ protocol securely realizes \mathcal{F} using random oracles and static corruptions when DL and DDH assumptions hold if the CL signature [Camenisch and Lysyanskaya, 2004] and the BLS signature [Boneh et al., 2003a] are unforgeable.*

Proof of Theorem 7.1: We first present a high-level description of our proof that consists of a sequence of games starting with the real-world protocol execution in Game 1. Moving on to 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: an ideal functionality \mathcal{F} and a simulator \mathcal{S} that simulates the real-world parties. Initially, we start with an “empty” functionality \mathcal{F} . With each game, we gradually change \mathcal{F} and update \mathcal{S} accordingly, moving from the real world to the ideal world, and culminating in the full PRIVÉ \mathcal{F} being realized as part of the ideal world, thus proving our proposed security model presented in § 6.2. The endmost goal of our proof is to prove the indistinguishability between Game 1 and Game 14, i.e., between the complete real world and the fully functional ideal world. This is done by proving that each game is indistinguishable from the previous one, starting from Game 1 and reaching Game 14.

Our proof of Theorem 7.1 starts with setting up the real-world games (Game 1 and Game 2), then introducing the ideal functionality in Game 3. At this stage, the ideal functionality \mathcal{F} only forwards its inputs to the simulator and simulates the real world. From Game 4 onward, \mathcal{F} starts executing the setup interface on behalf of the Issuer. Moving on to Game 5, \mathcal{F} handles simple verification and linking checks without performing any detailed checks at this stage; i.e., it only checks if the device belongs to a revocation list separately. In Games 6 – 7, \mathcal{F} executes the Join interface while performing checks to maintain the registered keys’ consistency. It also adds checks that allow only the devices that have successfully been enrolled to create signatures. Game 8 proves the anonymity of PRIVÉ by letting \mathcal{F} handle the sign queries on behalf of honest devices by creating swarm attestation using freshly generated random keys instead of running the sign algorithm using the device’s signing key. At the end of this game, we prove that by relying on DDH and DL constructions, an external environment will notice no change from previous games where the real-world sign algorithm was executed as explained in our UC model (§ 6.2). The sequence of games is explained in Figure 3. We use the “ \approx ” sign to express games’ indistinguishability. \mathcal{F}_i and \mathcal{S}_i in Figure 3 represent the ideal functionality \mathcal{F} and the simulator \mathcal{S} , respectively, as defined in the i^{th} game for $3 \leq i \leq 14$. Next, we present a sketch UC proof for Theorem 7.1.

Proof. **Game 1 (Real World):** This is PRIVÉ.

Game 2 (Transition to the Ideal World): An entity C is introduced. C receives all inputs from the honest parties and simulates the real-world protocol for

them. This is equivalent to Game 1, as this change is invisible to \mathcal{E} .

Game 3 (Transition to the Ideal World with Different Structure): We now split C into two parts, \mathcal{F} and \mathcal{S} , where \mathcal{F} behaves as an ideal functionality. It receives all the inputs and forwards them to \mathcal{S} , which simulates the real-world protocol for honest parties and sends the outputs to \mathcal{F} . \mathcal{F} then forwards these outputs to \mathcal{E} . This game is essentially equivalent to Game 2 with a different structure.

Game 4 (\mathcal{F} handles the setup): \mathcal{F} now behaves differently in the setup interface, as it stores the algorithms defined in Section 6.2. \mathcal{F} also performs checks and ensures that the structure of sid , which represents the issuer’s unique session identifier for an honest I , aborts if not. When I is honest, \mathcal{S} will start simulating it. Since \mathcal{S} is now running I , it knows its secret key. In case I is corrupt, \mathcal{S} extracts I ’s secret key from π_{ipk} and proceeds to the setup interface on behalf of I . By the simulation soundness of π_{ipk} , this game transition is indistinguishable for the adversary (Game 4 \approx Game 3).

Game 5 (\mathcal{F} handles the verification and linking): \mathcal{F} now performs the verification and linking checks instead of forwarding them to \mathcal{S} . There are no protocol messages; the outputs are exactly as in the real-world protocol. However, the only difference is that the verification algorithms that \mathcal{F} uses (namely ver_{DAA} and ver_{IoT}) do not contain revocation checks, so knowing KRL and IoTKRL for corrupt edge and IoT, respectively, \mathcal{F} performs these checks separately, and the outcomes are equal (Game 5 \approx Game 4).

Game 6 (\mathcal{F} handles the join): The join interface of \mathcal{F} is now changed. Specifically, \mathcal{F} stores the members that joined in its records. If I is honest, then \mathcal{F} stores the secret keys tsk and x_L extracted from π^1 and π_L by \mathcal{S} for corrupt edge and IoT devices, respectively. \mathcal{F} sets the tracing key for each honest edge device (as it already knows its key tsk) or calculates it from the extracted tsk (for corrupt edge devices). Only if the edge or the IoT device is already registered in EdgeMembers or IoTMembers , \mathcal{F} will abort the protocol. However, I has already tested this case before continuing with the query JOINPROCEED ; thus, \mathcal{F} will not abort. Knowing tsk and all children dsk s, \mathcal{F} proceeds with adding the children IoT devices into IoTMembers on behalf of edge devices. Therefore, \mathcal{F} and \mathcal{S} can interact to simulate the real protocol in all cases. Due to the simulation soundness of π^1 and π_L , Game 6 \approx Game 5.

Game 7 (\mathcal{F} further handles the join): If I is honest, then \mathcal{F} only allows the devices that joined to sign. An honest edge device will always check whether it joined with a TC in the real-world protocol, so there



Figure 3: Overview of the UC PRIVÉ Proof

is no difference for honest edge devices. In the case that an honest \mathcal{M}_j performs a join protocol with a corrupt edge device E_j and an honest Issuer, \mathcal{S} will make a join query with \mathcal{F} to ensure that \mathcal{M}_j and E_j are in `EdgeMembers`. Also, only joined IoT devices with certified public keys can create signatures. The parent edge device will check this before verifying its children's IoT signatures. Therefore $\text{Game 7} \approx \text{Game 6}$.

Game 8 (\mathcal{F} handles the sign/ SP1: Anonymity) (Simulating an edge device without knowing its secret): In this game, we want to prove that an external environment cannot distinguish when \mathcal{F} internally handles the signing queries instead of merely forwarding them to \mathcal{S} for \mathcal{M}_j . Suppose an environment can distinguish a signature (created by an honest edge device with a secret signing key tsk) from a signature constructed by the same party but with a randomly chosen fresh tsk . Then, we can use that environment to break a Decisional Diffie–Hellman DDH instance. \mathcal{M}_j uses tsk to set $Q \leftarrow g_1^{tsk}$ in the JOIN protocol, creates the proofs π_1 in joining and π in signing, and to compute link tokens nym . In the simulation, we set $Q \leftarrow \alpha$ and simulate all proofs π_1 and π . For nym , the power over the random oracle is used: \mathcal{S} chooses $H_1(\text{bsn}) = g_1^r$ for $r \leftarrow \mathbb{Z}_q$, and sets $nym \leftarrow \alpha^r = H_1(\text{bsn})^{tsk}$ without knowing tsk and output a signature. Anonymity of the IoT devices is also achieved against Verifiers through the use of *entirely random pseudonyms* (s, o) that don't reveal the identities of the IoT devices. Thus, starting with an IoT pseudonym, \mathcal{V} cannot tell the identity of the IoT device that generated this pseudonym unless it collaborates with the edge device. An external environment cannot distinguish between an honest IoT signature and a signature by the same party but with a randomly chosen fresh dsk . Therefore, $\text{Game 8} \approx \text{Game 7}$.

Game 9 (SP2: Traceability): When storing a new tsk or dsk , \mathcal{F} checks if `CheckkeyHonest` = 1 or `CheckkeyCorrupt` = 1 for both keys. If the device is corrupt, \mathcal{F} checks that `CheckkeyCorrupt` = 1 for the keys tsk or/and dsk that the simulator extracted. This check prevents the adversary from choosing different keys. There exists only a single tsk for every valid signature where $\text{identify}_{\text{DAA}}(\sigma_{\text{DAA}}^j, \mu, \text{bsn}, tsk) = 1$, and only a single dsk for every valid sig-

nature where $\text{identify}_{\text{IoT}}((\sigma, m, dsk, (s, o))) = 1$ for each edge and device, respectively, thus this check will never fail. For keys of honest devices, \mathcal{F} verifies that `CheckkeyHonest` = 1 whenever it receives or creates a new key. With these checks, we avoid registering keys for which matching signatures already exist. Since keys for honest devices are chosen uniformly from an exponentially large group and each signature has exactly one matching key, the chance that a signature under that key already exists is negligible ($\text{Game 9} \approx \text{Game 8}$).

Game 10 (SP4: Correctness): In this game, \mathcal{F} checks that honestly generated signatures are always valid. This is true since the sig_{DAA} and sig_{IoT} algorithms always create signatures that pass through verification checks. Also, they satisfy $\text{identify}_{\text{DAA}}(tsk, \sigma_{\text{DAA}}, \mu, \text{bsn}) = 1$ and $\text{identify}_{\text{IoT}}((\sigma, m, dsk, (s, o))) = 1$. \mathcal{F} ensures, using its internal records `MemberList` and `DomainKeys`, that honest users aren't sharing the same secret key tsk or dsk ; this is reduced with a non-negligible probability of solving the DL problem. Assume that \mathcal{F} receives an instance $h \in G_1$ of the DL problem and must answer $\log_{g_1}(h)$. \mathcal{F} chooses an honest device and simulates its tasks using the unknown DL of h as its secret key. When a tsk/dsk matches one of this device's signatures in the revocation list, then this must be the discrete log of h , as there is only one tsk/dsk matching a signature ($\text{Game 10} \approx \text{Game 9}$).

Game 11 (SP5: Non-frameability): CHECK1 ensures that there are no multiple tsk or dsk values matching one signature. \mathcal{F} also checks, with the help of its internal key records `Members` and `DomainKeys`, that no other device already has a key that would match this newly generated signature. If this fails, we can solve the DL problem: We simulate a TC using the unknown discrete logarithm of the DL instance as tsk or dsk , as in the DDH reduction before. If a matching tsk or dsk is found, then we have a solution to the DL problem. Therefore, if solving the DL problem is computationally infeasible, $\text{Game 11} \approx \text{Game 10}$.

Game 12 (SP6: Unforgeability): CHECK3 is added to \mathcal{F} to prevent anyone from forging signatures by using honest tsk or dsk and their credentials. If the Issuer is honest, CHECK2 prevents signing with in-

valid join credentials. This property is built on the unforgeability of the CL [Camenisch et al., 2016], combined with the unforgeability of the aggregate signatures [Boneh et al., 2003b].

Game 13 (SP3: Revocation) CHECK4 is added to \mathcal{F} . This ensures that honest devices are not being revoked. If an honest device is simulated, when a matching key is identified in KRL or IoTKRL, it must be the key of the target instance. Equivalent to solving the DL of the problem (Game 13 \approx Game 12).

Game 14 (SP7: Linkability): All the remaining checks of the ideal functionality \mathcal{F} related to link queries are now included. Since tsk , dsk only match one signature and no other signature, Game 14 is indistinguishable from Game 13, and \mathcal{F} now includes all the functionalities. This concludes the proof. \square

8 EVALUATION

We analytically evaluate the computational complexity and performance of our protocol by measuring the execution time of the core phases described in § 5.1. These can be divided into (i) offline, i.e., the operations which can be pre-computed or do not need to be executed in real-time (such as the Setup and Join phases described in § 5.1.1) and (ii) on-line operations. Since we are interested in the evaluation of PRIVÉ during runtime, the experiments presented here focus on the online operations (described in § 5.1.2 and § 5.1.3).

Evaluation Environment Setup & Testing

Methods: To better capture the resource-constrained nature of real-world environments, each edge device was emulated by a Raspberry Pi 4 (ARM v8), representing a node with medium computational power and storage capacity. Additionally, each edge device is equipped with an Infineon Optiga™ SLB 9670 TPM 2.0, as the underlying Root-of-Trust, leveraging the BN_P256 elliptic curve for all ECC-oriented crypto operations. The same platform was also used for evaluating the verification process of the attestation claims¹. Each IoT node was represented by the following platforms: Raspberry Pi Pico (ARM Cortex M0+) and Arduino Nano (ESP32-S3), demonstrating PRIVÉ’s capability to operate effectively on resource-constrained embedded devices.

(IoT) Signature Construction: Running the protocol on IoT devices consists of two sequential steps: (i) hash computation (for the construction of the attestation claims comprising the necessary trustworthi-

¹Please note that in practice, the Verifier can be a more powerful platform, resulting in improved verification timings

Table 2: Overhead of PRIVÉ on the IoT devices during signing

		Raspberry Pi Pico		Arduino Nano	
		Codebase (bytes)	Mean (ms) \pm (95% CI)	Mean (ms) \pm (95% CI)	
SHA-256	2048	2.644	0.009	1.056	0.003
	4096	5.158	0.008	2.007	0.002
SHA3	2048	5.210	0.017	2.037	0.003
	4096	10.019	0.018	3.864	0.003
Hash to point		70.846	0.042	14.474	0.006
Point multiplication		140.719	0.529	32.011	0.109

Table 3: Aggregation and verification of IoT signatures

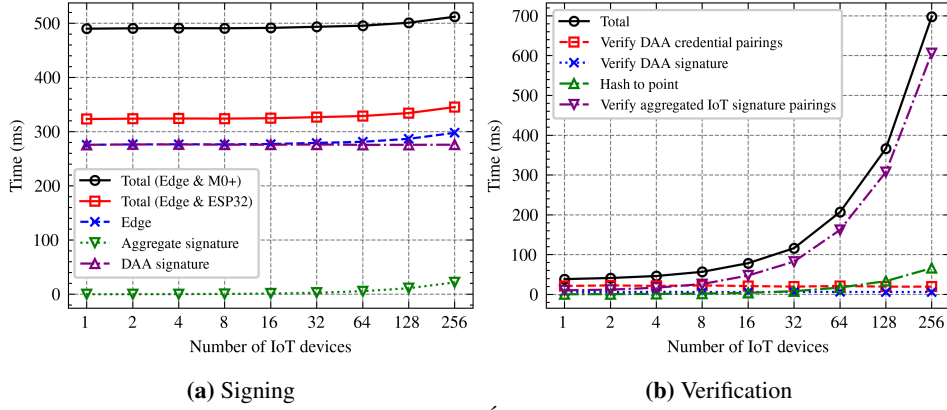
No. IoT	Aggregation (ms)	Verification (ms)
1	-	10.455
2	0.104	12.017
4	0.291	16.885
8	0.630	26.065
16	1.333	47.494
32	2.933	82.484
64	5.663	161.994
128	11.081	307.348
256	21.885	606.069

ness evidence), and (ii) signature computation (hash-to-point and point multiplication). Tab. 2 summarizes the respective computational costs. Our experiments highlight the efficiency of the (configuration) integrity checks even for a rather large codebase (≈ 4 KB) (in real-world IoT actuators the average code size is ≈ 2 KB). As it will be described later on, the online operations performed by the IoT device incur significantly lower overhead on the overall protocol lifecycle compared to the tasks of the attestation process executed on the more resource-rich edge device.

Aggregation (Edge) and Verification (V): The time required for the aggregation of IoT signed at-

Table 4: DAA SIGN Operation Timing (HW-TPM)

Activity	Mean (ms)	\pm (95% CI)
Total Application Stack	609.05	0.57/0.55
TPM2_StartAuthSession	23.93	2.67
TPM2_PolicyCommandCode	14.99	0.03
TPM2_PolicyOR	21.64	0.09
TPM2_LoadExternal	71.34	0.24/0.44
TPM2_VerifySignature	55.97	2.64
TPM2_PolicyAuthorize	34.36	7.74
TPM2_Commit	95.11	0.03/0.05
TPM2_Hash	34.04	0.04/0.009
TPM2_StartAuthSession	23.19	2.67
TPM2_PolicyNV	20.99	0.03
TPM2_PolicyOR	40.62	0.09
TPM2_VerifySignature	53.92	2.64
TPM2_PolicyAuthorize	34.36	7.41
TPM2_Sign	39.77	0.04/0.01
TPM2_FlushContext	13.27	0.02/0.01



(a) Signing (b) Verification
Figure 4: Scalability of PRIVÉ under trusted Edge devices

testation claims (by the edge device), and their subsequent verification by \mathcal{V} , is summarized in Tab. 3, for a varying space of IoT signatures. What we can observe is that the complexity of the aggregated signature construction is nearly linear to the number of signatures. This is crucial to the overall protocol serving as evidence to its low complexity; i.e., does not increase (e.g., by growing exponentially) with the number of signatures, making the aggregation operation efficient even for a large size of IoT devices. In Figs. 4a and 4b, we also capture the device-level overhead during the attestation and verification phases under a trusted edge setting; i.e., without key restriction policies, which necessitates the execution of additional commands for asserting the correctness of the attestation key binding.

In this context, the most performance-intensive operation is the verification of the validity of each edge device DAA signature on the aggregate result w.r.t. to the basename leveraged (dictating the anonymity level). This is also then followed by the validation of the overall aggregated IoT result (§ 4). For the former, the DAA_VERIFY is split into two operational blocks: verification of the (randomized) DAA credential that takes up $\approx 155ms$ (irrespective of whether (bsn) changes for every signature or not) and verification of the ECDSA signature that takes up $\approx 20ms$. The DAA Credential verification, while efficient, sums up to a higher required time than a simple verification, as it also includes the execution of additional TPM2_NV_Read and TPM2_PolicyNV commands for asserting that the credential’s associated private DAA Key is binded to the correct key restriction usage policy holding the policy hash of the expected (correct) device configuration integrity. For the latter, the verification of the aggregated IoT signature can be split into a “hash-to-point” step to compute $H(m_k)$, and the computations of the pairings to perform the actual IoT signature verification. This slow verifica-

tion time is mainly due to the complexity of the pairing operation, which is very expensive w.r.t. elliptic curve point additions used for the aggregation.

DAA Signature (Edge): In Tab. 4, we present the timings of the DAA_SIGN operation (performed by edge devices) in a zero-trust setting. Although the overall signing takes $\approx 600ms$, it should be highlighted that the actual signing operation (TPM2_Sign command) is quite fast, requiring $\approx 40ms$. However, recall that in the zero-trust setting, the DAA key is protected by key restriction usage policies, so it can only be used if the device is in a correct state (and, thus, providing implicit attestation of the edge device). As aforementioned, this is captured by the execution of TPM2_NV_Read and TPM2_PolicyNV commands, followed by the TPM2_PolicyOR and TPM2_PolicyAuthorize commands, which not only verify the correct key binding (otherwise, the signature will not have been constructed correctly if the current device state does not match with the trusted state been stored on the Platform Configuration Registers) but also enable the ratification of the pseudonyms’ validity used by the IoT devices; i.e., not have been revoked prior to be used for signing. This process takes $\approx 100ms$. Also, note that the duration of $95.11ms$ for the TPM2_Commit command refers to the case where linkability is not needed, therefore a basename is not used. In case linkability capabilities are needed, a basename needs to be used, which increases the execution time of this command to $224ms$, thus introducing a trade-off in terms of efficiency versus linkability. However, even in this case, the time needed for tracing back to the source of a failed attestation signature, irrespectively of the *size* and *depth* of the swarm topology, is rather efficient - serving as evidence on the applicability of PRIVÉ in safety-critical applications where pinpointing possible indications of risk is crucial; i.e., enabling the fast deployment of reaction strategies.

9 CONCLUSIONS

In this paper, we provided a novel design for a swarm attestation protocol, with the innovative feature of identity privacy preservation and accountable attestation, by creating an enhanced version of a DAA variant with traceability. Our solution avoids the limitations of existing schemes by providing the ability to conceal the identity of the devices (unless they are compromised) and the ability to trace a failed attestation to the source swarm device that caused the failure. Additionally, to the best of our knowledge, we provide the first security analysis for swarm attestation with enhanced privacy features in the UC model. Future work will explore how PRIVÉ can be leveraged to achieve attestation evidence privacy, leading us to complete Zero Trust architectures.

ACKNOWLEDGMENT

This research has received funding from the European Union's Horizon Europe EU Research & Innovation programs ENTRUST and REWIRE under Grant Agreement No. 101095634 and 101070627, respectively.

REFERENCES

- (2016). Trusted Platform Module Library Part 1: Architecture. Standard, Trusted Computing Group (TCG).
- (2018). IEEE standard for adoption of OpenFog reference architecture for fog computing. *IEEE Std 1934-2018*.
- Abera, T., Brasser, F., Jauernig, P., Koisser, D., and Sadeghi, A.-R. (2021). Granddetauto: Detecting malicious nodes in large-scale autonomous networks. *RAID '21*.
- Ambrosin, M., Conti, M., Ibrahim, A., Neven, G., Sadeghi, A.-R., and Schunter, M. (2016). SANA: Secure and Scalable Aggregate Network Attestation. In *CCS '16*.
- Ambrosin, M., Conti, M., Lazzeretti, R., Rabbani, M. M., and Ranise, S. (2020). Collective remote attestation at the internet of things scale: State-of-the-art and future challenges. *IEEE Comm. Survey*, 22(4).
- Ammar, M., Crispo, B., and Tsudik, G. (2020). Simple: A remote attestation approach for resource-constrained iot devices. In *ACM/IEEE on Cyber-Physical Systems*.
- Asokan, N., Brasser, F., Ibrahim, A., Sadeghi, A.-R., Schunter, M., Tsudik, G., and Wachsmann, C. (2015). Seda: Scalable embedded device attestation. *CCS '15*.
- Boneh, D. (1998). The decision diffie-hellman problem. In *International algorithmic number theory symposium*, pages 48–63. Springer.
- Boneh, D., Gentry, C., Lynn, B., and Shacham, H. (2003a). A Survey of Two Signature Aggregation Techniques. *CryptoBytes*, 6.
- Boneh, D., Gentry, C., Lynn, B., and Shacham, H. (2003b). Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *EUROCRYPT 2003*, pages 416–432.
- Brickell, E., Camenisch, J., and Chen, L. (2004). Direct Anonymous Attestation. *CCS '04*, pages 132–145.
- Brickell, E., Chen, L., and Li, J. (2008). A New Direct Anonymous Attestation Scheme from Bilinear Maps. In *Trust 2008*, page 166–178.
- Camenisch, J., Chen, L., Drijvers, M., Lehmann, A., Novick, D., and Urian, R. (2017). One tpm to bind them all: Fixing tpm 2.0 for provably secure anonymous attestation. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 901–920.
- Camenisch, J., Drijvers, M., and Lehmann, A. (2016). Universally Composable Direct Anonymous Attestation. In *PKC*, volume 9615 of *LNCS*, pages 234–264.
- Camenisch, J. and Lysyanskaya, A. (2004). Signature schemes and anonymous credentials from bilinear maps. In *International cryptology Conf*. Springer.
- Carpent, X., ElDefrawy, K., Rattanavipan, N., and Tsudik, G. (2017). Lightweight swarm attestation: A tale of two lisa-s. *ASIA CCS '17*.
- Chen, L., Dong, C., El Kassem, N., Newton, C. J., and Wang, Y. (2023). Hash-based direct anonymous attestation. In *International Conference on Post-Quantum Cryptography*, pages 565–600. Springer.
- Chen, L., El Kassem, N., Lehmann, A., and Lyubashevsky, V. (2019). A framework for efficient lattice-based daa. In *CYSARM*, pages 23–34.
- Chen, L., El Kassem, N., and Newton, C. J. (2024). How to bind a tpm's attestation keys with its endorsement key. *The Computer Journal*, 67(3):988–1004.
- Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Trans. on Inf. Theory*, 29(2).
- Dushku, E., Rabbani, M. M., Vliegen, J., Braeken, A., and Mentens, N. (2023). Prove: Provable remote attestation for public verifiability. *Journal of Information Security and Applications*, 75:103448.
- El Kassem, N., Chen, L., El Bansarkhani, R., El Kaafarani, A., Camenisch, J., Hough, P., Martins, P., and Sousa, L. (2019). More efficient, provably-secure direct anonymous attestation from lattices. *Future Generation Computer Systems*, 99:425–458.
- Kohnhäuser, F., Büscher, N., and Katzenbeisser, S. (2018). Salad: Secure and lightweight attestation of highly dynamic and disruptive networks. *ASIACCS '18*.
- Larsen, B., Giannetos, T., Krontiris, I., and Goldman, K. (2021). Direct anonymous attestation on the road: Efficient and privacy-preserving revocation in c-its. *WiSec '21*, page 48–59.
- Le-Papin, J., Dongol, B., Treharne, H., and Wesemeyer, S. (2023). Verifying list swarm attestation protocols. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.
- McCurley, K. S. (1990). The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42. USA.
- Meier, S., Schmidt, B., Cremers, C., and Basin, D. (2013). The tamarin prover for the symbolic analysis of security protocols. In *Conf. on Comp. Aided Verification*.
- Wesemeyer, S. and all (2020). Formal analysis and implementation of a tpm 2.0-based direct anonymous attestation scheme. *ASIA CCS '20*.