OASIS: An Organizational CertificateLess Aggregate Signature Scheme in Distributed Networks for IoT

Clémentine Gritti

OASIS: An Organizational CertificateLess Aggregate Signature Scheme in Distributed Networks for IoT

Clémentine Gritti

October 21, 2024

Abstract

In recent years, a large number of CertificateLess Aggregate Signature (CLAS) schemes have been proposed to overcome both the complexity of Public Key Infrastructure (PKI) certificate management and the key escrow problem. These CLAS schemes have mostly been developed for the Internet of Things (IoT). However, the current CLAS schemes require the trusted authority to manage all the devices in a network, whose number and turn-over are huge.
One way to alleviate devices' management in IoT while improving access to resources is to consider a distributed architecture. In this paper, we introduce **OASIS**, an **O**rganizational CertificateLess **A**ggregate **SI**gnature **S**cheme in IoT networks. OASIS is a hierarchical CLAS scheme that delegates the devices' management workload to multiple entities, while mitigating PKI certification and key escrow issues. We prove the security of OASIS in the random oracle model. Furthermore, the experimental results show that OASIS is well suitable for IoT distributed systems.

Keywords: Certificateless aggregate signatures, Organizational chart, Distributed networks, Internet of Things, Random oracle model.

1 Introduction

A distributed network has its components and data depend on multiple sources. Such a network configuration allows every entity to communicate with one another without going through a centralized point. In particular, a distributed network is a collection of multiple, independently run networks that are collectively managed. In this paper, we designate those independent networks as *sub-networks*.

Over the last few years, distributed computing has been seen as a beneficial technology for the Internet of Things (IoT) to improve its security, scalability and efficiency. IoT connects devices to networks through information sensing equipment. Devices collect and exchange data to realize intelligent identification, positioning, tracking, supervision and other functions. For instance, devices share and process medical data in Healthcare Wireless Medical Sensor Networks (HWMSNs) to improve patients and practitioners' experiences [Kumar and Lee(2012)].

1.1 Problem Statement

With the rapid development of IoT technologies, many practical applications have been developed to serve individuals' daily lives, such as wireless medical care monitoring [Saeed et al.(2018)]. Distributed networks in IoT have brought a lot of convenience to companies and individuals; however, they expand in size and complexity rapidly. Consequently, maintaining their performance and availability has become increasingly difficult. Challenges include security, data consistency, sub-network latency, and resource allocation.

Let us illustrate those limitations with a use case: wireless sensor networks, that have become an omnipresent application in healthcare [Ko et al.(2010)]. This emerging technology enables healthcare entities (e.g. patients, practitioners, hospitals) to improve and grow the quality and efficacy of medical treatments and processes. HWMSNs aim to offer real-time medical information transfer, reliable patient-practitioner communication, patient mobility and energy-efficient routing. Online, instant data sharing in healthcare improves the efficiency and availability of medical care. Nevertheless, deploying this recent technology at a large scale in distributed systems without posing the security concerns impacts the integrity of highly sensitive medical information [Kumar and Lee(2012)]. The huge number of devices in a medical distributed network may impede the security if they collect and exchange altered data. Technical challenges encountered in HWMSNs include the constrained resources of medical sensing devices (e.g. storage, bandwidth, power consumption), impacting the quality of medical service and the interoperability between devices in the network [de Schatz et al.(2012)]. Therefore, medical data privacy, and thus patients' safety, which are essential requirements of healthcare applications, must be considered carefully based on such constraints.

Security in distributed systems must consider communication mechanisms among entities (e.g. entity authentication, and data integrity and confidentiality) and access control to system resources [Ameen et al.(2012)]. As seen above, based on incomplete and/or altered medical information, a practitioner may make an unwise or erroneous diagnosis to a patient, putting her life in danger [Zhang et al.(2019)]. Digital signatures are a mechanism to achieve data integrity, thus preventing and detecting unauthorized modification of sensitive data.

A distributed system is deployed over the Internet, thus requires a strong Public Key Infrastructure (PKI). In a PKI, entities' public keys are authenticated by certification authorities through certificates. However, the management of certificates is complicated, especially when a distributed network involves many devices with a high turn-over. An extension of digital signature schemes uses the identity of the entities to generate their keys. The advantage is that verifying the signature of a signer only requires her identity, rather than her public key certified by a certification authority. Nevertheless, identity-based schemes suffer from the key escrow problem as keys are generated by a unique trusted *Key Generation Center* (KGC).

Al-Riyami and Paterson [Al-Riyami and Paterson(2003)] introduced CertificateLess Public Key Cryptography (CL-PKC) to overcome heavy certificate management in PKI and key escrow problem in identity-based schemes. CL-PKC uses identities of entities to create their key pairs (as in identity-based

schemes) while reducing the trust on the KGC by letting the entities generate their own secret value necessary for signing messages. In a CertificateLess Signature (CLS) scheme, the KGC is responsible of registering the devices embedded in the network by generating their partial signing keys. Each device also creates their own secret value. The device needs both the partial signing key and secret value to sign a message. Nevertheless, the CLS scheme in [Al-Riyami and Paterson(2003)] suits environments with few participating entities, where a verifier can easily check signatures of signers one by one. Such an assumption must not apply to distributed IoT networks, where the number of devices is too high, and so the verifier's workload.

One variant of CLS is CertificateLess Aggregate Signature (CLAS) [Gong et al.(2007), Au et al.(2007)]. The difference between CLS and CLAS comes when verifying the signatures. In CLAS schemes, the signatures are aggregated, resulting into one global signature. Therefore, the verifier only needs to check one signature for the whole group of signers. Such a design allows to carefully address the technical challenges encountered in expanded distributed architectures. Nevertheless, there remains one issue with the KGC dealing alone with a huge number of devices connected to a network and managing all their keys directly.

1.2 Idea

We introduce **OASIS**, an **O**rganizational CertificateLess **A**ggregate **SI**gnature **S**cheme in distributed networks for IoT. To alleviate KGC's management workload, we distribute a network into smaller *sub-networks*, such that each of them is managed by a gateway connected to the KGC and devices are grouped into different sub-networks. This creates a 2-level hierarchy as for an organizational chart. The KGC (root) is now responsible of generating the secret key of each sub-network's gateway. Then, each gateway (intermediate level) generates the partial signing key of the devices connected to it, using both its secret key and a secret value picked at random by the gateway itself. The device (bottom level) signs a message using both the partial signing key and a secret value picked at random by the device itself. Since both the gateway and devices have their own secret values, key escrow issues are overcome at all levels in the distributed network. As in traditional CLAS schemes, individual signatures are aggregated. Verification of individual signatures and aggregate signatures is a public process.

In addition to improve the PKI management, this organizational design allows to better control unfortunate events at devices' level. Let us suppose that a corrupted device aims to infect the network. With a traditional design, the whole network can be a victim of the attack. However, with our hierarchical setting, only the sub-network with the corrupted device may suffer from the attack, while the rest of the network can remain as normal. Indeed, let's the KGC decide to discard the connection with the attacked sub-network by, for example, revoking the corresponding gateway, while still manages the remaining sub-networks that have not been subject to the attack.

Our scheme extends Gritti et al.'s 2-level Identity-Based Aggregate Signature (2-IBAS) scheme [Gritti et al.(2018b)] by embedding techniques from [Al-Riyami and Paterson(2003)] to mitigate KGC's workload and key escrow problem. 2-IBAS was developed specifically for IoT by taking into account the huge number of devices in networks, their heterogeneity in terms of provenance and design, and their limited resources in terms of communication, computation

and storage. However, this scheme suffers from the key escrow problem since devices must solely rely on a trusted entity to obtain their signing keys. Instead, we enable devices to generate their own secret value, that is used as an input for signature generation in addition to their partial signing key delivered by their gateway. Similarly, the gateway needs two secret keys, one from the KGC and one from itself, to generate the partial signing keys of devices connected to it. We prove that our scheme is secure regarding Type-I and Type-II adversaries in the random oracle model [Al-Riyami and Paterson(2003)]. We also verify that our scheme is realistically deployable in distributed networks such as for IoT.

1.3 Related Work

To eliminate the use of certificates and to prevent the key escrow problem, Al-Riyami and Paterson [Al-Riyami and Paterson(2003)] introduced the concept of CL-PKC and proposed the first CLS scheme. Each entity owns two secret keys: one is generated by the KGC and one is generated by the entity itself. Both keys are needed to generate a signature on a message. Nevertheless, Huang et al. [Huang et al.(2005)] described an attacker that can successfully forge a certificateless signature in Al-Riyami and Paterson's security model. The authors proposed a new scheme to fix this problem. Boneh et al. [Boneh et al.(2003)] presented an Aggregate Signature (AS) scheme. In this scheme, the signatures, on different messages and from various signers, are collected and aggregated, resulting into one unique, global signature. The verifier only needs to check the latter to validate all the signatures. Such a design greatly reduces the workload at the verifier's side. Subsequently, multiple schemes have been proposed applicable to IoT environments, such as AS schemes [Shen et al.(2018)], identity-based AS schemes [Shen et al.(2017)], CLS schemes [Huang et al.(2005), Au et al.(2007)] and CLAS schemes [Gong et al.(2007), Zhang and Zhang(2009), He et al.(2013)]. Recently, other CLAS proposals have been released [Shen et al.(2019), Kamil and Ogundoyin(2019), Cui et al.(2018), Kumar et al.(2018), Gayathri et al.(2019), Y. Zhan and Lu(2021), Yang et al.(2021), Thumbur et al.(2021)], with the aim of finding a good trade-off between efficiency and security based on the technical IoT constraints. Most of the papers suggested pairing-free CLASs, since pairing operations are noticeably costly and thus not suitable for resource-constrained devices. However, We et al. [Wu et al.(2018)] found out that Kumar et al.'s CLAS scheme [Kumar et al.(2018)] is vulnerable to a honest-but-curious KGC. They then suggested a better version of the CLAS scheme. Moreover, Liu et al. [Liu et al.(2020)] pointed out that Gayathri et al.'s scheme [Gayathri et al.(2019)] is not secure. They proposed an improved pairing-free CLAS scheme. Nevertheless, all the aforementioned CLAS solutions do not consider an organizational chart in the network. Indeed, the KGC is directly responsible of all the devices, by generating their partial signing key, rather than the gateway of the sub-network in which devices are installed. When a distributed IoT network comprises a huge number of devices, such as an HWMSN, a 2-level hierarchy allows a better and easier management of devices and their keys at the KGC's side.

2 OASIS Definition

2.1 Use Case

First, let's imagine OASIS in Healthcare Wireless Medical Sensor Networks (HWMSNs). The KGC is in charge of the network of a hospital and of the sub-networks. Each sub-network is led by a gateway and contains multiple devices connected to the gateway. For instance, the sub-network is a medical room equipped with sensing devices that monitor various elements of the patient (e.g. heart pulsation, glucose level). Each device collects raw data at regular time intervals and submits it to the gateway. To guarantee that the collected data has not been tampered in transit, the device signs the data. The gateway collects the signature/data pairs and aggregates the signatures to obtain one global signature representing the sub-network. This allows to reduce the data traffic: instead of having a huge number of devices' individual signatures being broadcast over the whole network, only few aggregate signatures from the gateways of sub-networks are actually submitted to the KGC. A verifier (public entity) checks the validity of each sub-network's signature. We depict our use case in Figure 1.

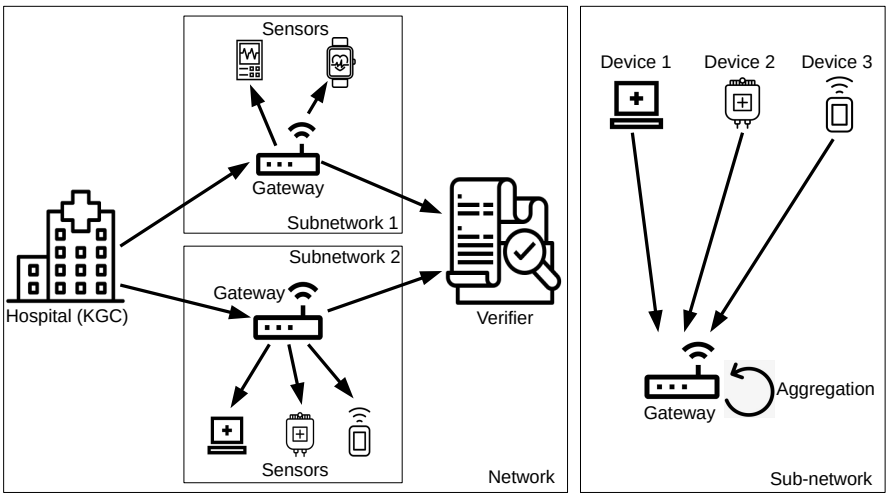

Figure 1: [LEFT] A hospital (KGC) comprises 2 sub-networks (gateways). A gateway registers its sensors and forwards the aggregate signatures to the verifier. [RIGHT] Each device in a sub-network generates a signature and forwards it to its gateway. The gateway collects individual signatures and aggregates them.

2.2 Overview

Let us describe the four entities involved in OASIS:
• The *KGC* is a single trusted authority per IoT distributed network. The KGC identifies and registers the gateways in the network by delivering to each of them a secret key through a secure communication channel.

• The *gateway* receives a secret key from the KGC and creates a secret value by itself. From this secret value, a public key is generated and made available to the network. Using both the secret key and secret value, it identifies devices that are installed in its sub-network and delivers their partial signing keys. It also collects the signatures of all the devices in its sub-network. It optionally checks each signature individually. Once all validity checks pass, the gateway aggregates the signatures, resulting into one global signature.
• The *device* receives a partial signing key from the gateway it is connected to. It also generates a secret value by itself. From this secret value, a public key is generated and made available to the network. It must uses both the partial signing key and secret value to sign a message. The device is expected to create one signature per round, by embedding a counter in its signature. The public key is used to verify the signature.
• The *verifier* verifies the validity of the aggregate signature given the messages of devices, the public keys of the devices, and the public key of the gateway. If the validity check passes, then it means that the data collected by its devices have not encountered any modification in transit. The verification is made public.

2.3 Formal Definition

Let us describe the algorithms of OASIS:
• $\mathsf{Setup}(\lambda) \rightarrow (params, msk)$. On input a security parameter λ, the Setup algorithm, run by the KGC, outputs the public parameters $params$ and the master secret key msk.
• $\mathsf{KeyGen}_{gat}(params, msk, I_i) \rightarrow sk_i$. On inputs the public parameters $params$, the master secret key msk and the identity I_i of the gateway, the KeyGen_{gat} algorithm, run by the KGC, outputs the secret key sk_i of the gateway. The key is sent to the gateway over a secure channel.
• $\mathsf{SecretGen}_{gat}(params) \rightarrow \beta_i$. On inputs the public parameters $params$, the $\mathsf{SecretGen}_{gat}$ algorithm, run by the gateway, outputs the secret value β_i. The value is securely kept by the gateway.
• $\mathsf{PubKeyGen}_{gat}(params, \beta_i) \rightarrow pk_i$. On inputs the public parameters $params$ and the secret value β_i of the gateway, the algorithm $\mathsf{PubKeyGen}_{gat}$, run by the gateway, outputs the public key pk_i.
• $\mathsf{PartKeyGen}_{dev}(params, sk_i, \beta_i, I_{i,j}) \rightarrow psk_{i,j}$. On inputs the public parameters $params$, the secret key sk_i and secret value β_i of the gateway, and the identity $I_{i,j}$ of the device connected to the gateway, the $\mathsf{PartKeyGen}_{dev}$ algorithm, run by the gateway, outputs a partial signing key $psk_{i,j}$. The key is sent to the device over a secure channel.
• $\mathsf{SecretGen}_{dev}(params) \rightarrow x_{i,j}$. On inputs the public parameters $params$, the $\mathsf{SecretGen}_{dev}$ algorithm, run by the device, outputs the secret value $x_{i,j}$. The value is securely kept by the device.
• $\mathsf{KeyGen}_{dev}(params, psk_{i,j}, x_{i,j}) \rightarrow sk_{i,j}$. On inputs the public parameters $params$, the partial signing key $psk_{i,j}$ and secret value $x_{i,j}$ of the device, the KeyGen_{dev} algorithm, run by the device, outputs the complete signing key $sk_{i,j}$.
• $\mathsf{PubKeyGen}_{dev}(params, x_{i,j}, pk_i) \rightarrow pk_{i,j}$. On inputs the public parameters $params$, the secret value $x_{i,j}$ of the device and the public key pk_i of the gateway, the $\mathsf{PubKeyGen}_{dev}$ algorithm, run by the device, outputs the public key $pk_{i,j}$.

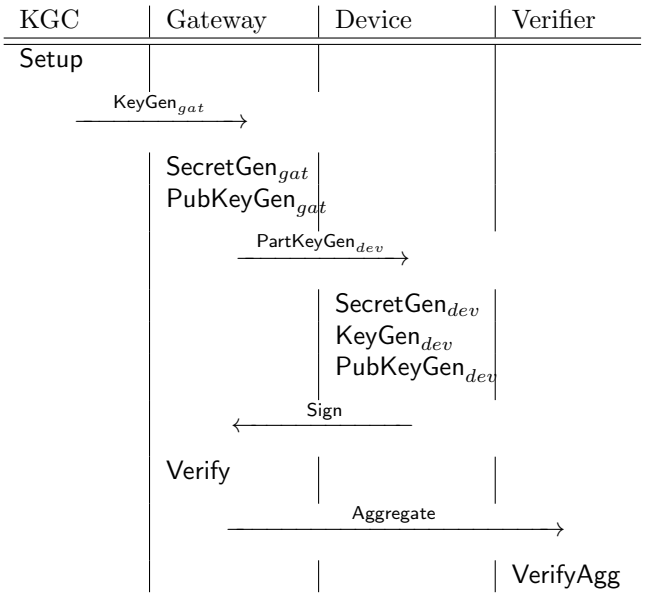

Table 1: Flow chart of OASIS. Arrows link a source entity which runs the algorithm with the appropriate inputs with a destination entity which receives the output.

• $\mathsf{Sign}(params, sk_{i,j}, m_j, cnt) \rightarrow \sigma_{i,j}$. On inputs the public parameters $params$, the signing key $sk_{i,j}$ of the device, a message m_j and a fresh counter cnt, the Sign algorithm, run by the device, outputs a signature $\sigma_{i,j}$ on message m_j.
• $\mathsf{Aggregate}(params, \{\sigma_{i,j}\}_{j\in[1,l]}) \rightarrow \sigma_i$. On inputs the public parameters $params$, the set $\{\sigma_{i,j}\}_{j\in[1,l]}$ of all signatures of devices connected to the gateway, the Aggregate algorithm is run by the gateway as follows.
Optionally, the gateway first runs the algorithm Verify to check that each signature $\sigma_{i,j}$ is valid given the message m_j. If at least one signature is not valid, then the gateway aborts. Otherwise, it proceeds by aggregating all the signatures in $\{\sigma_{i,j}\}_{j\in[1,l]}$ to obtain the aggregate signature σ_i.
• $\mathsf{Verify}(params, I_i, I_{i,j}, \sigma_{i,j}, m_j, pk_{i,j}) \rightarrow \{\text{"Accept"}, \text{"Reject"}\}$. On inputs the public parameters $params$, the identity I_i of the gateway, the identity $I_{i,j}$ of the device, the signature $\sigma_{i,j}$, the message m_j and the public key $pk_{i,j}$ of the device, the algorithm Verify, run by the gateway before aggregation, outputs either "Accept", i.e. $\sigma_{i,j}$ is a valid signature for the message m_j, or "Reject", i.e. $\sigma_{i,j}$ is not valid.
• $\mathsf{VerifyAgg}(params, I_i, \{I_{i,j}\}_{j\in[1,l]}, \sigma_i, \{m_j\}_{j\in[1,l]}, \{pk_{i,j}\}_{j\in[1,l]}) \rightarrow \{\text{"Accept"}, \text{"Reject"}\}$. On inputs the public parameters $params$, the identity I_i of the gateway, the set of identities $\{I_{i,j}\}_{j\in[1,l]}$ of the devices connected to the gateway, the aggregate signature σ_i, the set of messages $\{m_j\}_{j\in[1,l]}$ and the set of public keys $\{pk_{i,j}\}_{j\in[1,l]}$ of the devices connected to the gateway, the algorithm VerifyAgg, run by the verifier, outputs either "Accept", i.e. σ_i is a valid aggregate signature for all message $\{m_j\}_{j\in[1,l]}$, or "Reject", i.e. σ_i is not valid.

We depict the flow chart of our proposed solution in Table 1.

2.4 Security Models

Correctness

Let us be given $(params, msk) \leftarrow \mathsf{Setup}(\lambda)$, the secret key $sk_i \leftarrow \mathsf{KeyGen}_{gat}(params, msk, I_i)$, the secret value $\beta_i \leftarrow \mathsf{SecretGen}_{gat}(params)$ and the public key $pk_i \leftarrow \mathsf{PubKeyGen}_{gat}(params, \beta_i)$ of the gateway with identity I_i, the partial keys $psk_{i,j} \leftarrow \mathsf{PartKeyGen}_{dev}(params, sk_i, \beta_i, I_{i,j})$, the secret values $x_{i,j} \leftarrow \mathsf{Secret-Gen}_{dev}(params)$, the signing keys $sk_{i,j} \leftarrow \mathsf{KeyGen}_{dev}(params, psk_{i,j}, x_{i,j})$ and the public keys $pk_{i,j} \leftarrow \mathsf{PubKeyGen}_{dev}(params, x_{i,j}, pk_i)$ of the devices with identity $I_{i,j}$ for $j \in [1, l]$, connected to the gateway with identity I_i. Let $\sigma_{i,j} \leftarrow \mathsf{Sign}(params, sk_{i,j}, m_j, cnt)$ be the signatures of the devices based on their respective messages m_j and the fresh counter cnt, for $j \in [1, l]$. Let $\sigma_i \leftarrow \mathsf{Aggregate}(params, \{\sigma_{i,j}\}_{j\in[1,l]})$ be the signature obtained from aggregating the signatures $\sigma_{i,j}$ for $j \in [1, l]$. For each $j \in [1, l]$, the algorithm $\mathsf{Verify}(params, I_i, I_{i,j}, \sigma_{i,j}, m_j, pk_{i,j})$ outputs "Accept". Moreover, the algorithm $\mathsf{Verify-Agg}(params, I_i, \{I_{i,j}\}_{j\in[1,l]}, \sigma_i, \{m_j\}_{j\in[1,l]}, \{pk_{i,j}\}_{j\in[1,l]})$ outputs "Accept".

OASIS also guarantees that individual and aggregate signatures are existentially unforgeable. We combine CLAS and 2-IBAS security models [Al-Riyami and Paterson(2003), Gritti et al.(2018b)] to embed the hierarchical structure of OASIS into our security models. This means that at least one entity is not corrupted at each hierarchical level, i.e. one gateway and one device respectively. We allow adversaries to extract secret/partial signing keys for identities of their choice. We define Game I and Game II for the adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. The honest-but-curious adversary $\mathcal{A}_1$ cannot access the master secret key msk but can replace public keys of any entity (gateway and device) with a value of its choice. The malicious adversary $\mathcal{A}_2$ can access the master secret key msk (held by the KGC) but cannot replace any public keys.

Game I

Game I between the challenger $\mathcal{C}_1$ and adversary $\mathcal{A}_1$ is defined as follows.

Setup: On input the security parameter λ, the challenger $\mathcal{C}_1$ runs the Setup algorithm to get the master secret key msk and the public parameters $params$. $\mathcal{C}_1$ sends $params$ to $\mathcal{A}_1$ and keeps msk.

Queries: The adversary $\mathcal{A}_1$ performs a polynomially bounded number of queries.

• *Secret/partial signing key queries:* $\mathcal{A}_1$ requests the secret/partial signing key of an entity with identity I (either a gateway or a device). $\mathcal{C}_1$ runs the appropriate key generation algorithm and sends the secret/partial signing key to $\mathcal{A}_1$.

• *Public key queries:* $\mathcal{A}_1$ requests the public key of an entity with identity I (either a gateway or a device). $\mathcal{C}_1$ runs the appropriate public key generation algorithm and sends the public key to $\mathcal{A}_1$.

• *Secret queries:* $\mathcal{A}_1$ requests the secret of an entity with identity I (either a gateway or a device). $\mathcal{C}_1$ runs the appropriate secret generation algorithm and sends the secret to $\mathcal{A}_1$.

• *Public key replacement queries:* $\mathcal{A}_1$ chooses a new public key pk' for an entity with identity I and $\mathcal{C}_1$ records such a replacement.

• *Signature queries:* $\mathcal{A}_1$ requests the signature on a message m. $\mathcal{C}_1$ runs the appropriate signature generation algorithm and sends the signature to $\mathcal{A}_1$.

Forgery: $\mathcal{A}_1$ outputs a set of l devices with identity $\{I^*_{i,j}\}$, connected to the same gateway with identity I^*_i, with the set of their corresponding public keys $\{pk^*_{i,j}\}$, a set of messages $\{m^*_j\}$, and an aggregate signature σ^*_i.

The adversary $\mathcal{A}_1$ wins the game if:

- σ^*_i is a valid aggregate signature.
- Secret/partial signing key queries and secret queries have never been made for at least one identity in the set $\{I^*_{i,j}\}$ and for I^*_i (i.e. there is at least one uncorrupted identity for each entity type). W.l.o.g., let us assume that such identities are I^*_i and $I^*_{i,1}$ (i.e. $j = 1$).
- Signature queries have never been made on $(I^*_i, I^*_{i,1}, m^*_1)$.

Game II

Game II between the challenger $\mathcal{C}_2$ and adversary $\mathcal{A}_2$ is defined as follows.

Setup: On input the security parameter λ, the challenger $\mathcal{C}_2$ runs the Setup algorithm to get the master secret key msk and the public parameters $params$. $\mathcal{C}_2$ sends $params$ and msk to $\mathcal{A}_2$.

Queries: The adversary $\mathcal{A}_2$ performs a polynomially bounded number of queries.

- *Partial signing key queries:* $\mathcal{A}_2$ requests the partial signing key of an entity with identity I (only a device as the adversary can generate itself the secret key of the gateway by using the master secret key msk). $\mathcal{C}_2$ runs the appropriate key generation algorithm and sends the partial signing key to $\mathcal{A}_2$.
- *Public key queries:* $\mathcal{A}_2$ requests the public key of an entity with identity I (either a gateway or a device). $\mathcal{C}_2$ runs the appropriate public key generation algorithm and sends the public key to $\mathcal{A}_2$.
- *Secret queries:* $\mathcal{A}_2$ requests the secret of an entity with identity I (either a gateway or a device). $\mathcal{C}_2$ runs the appropriate secret generation algorithm and sends the secret to $\mathcal{A}_2$.
- *Signature queries:* $\mathcal{A}_2$ requests the signature on a message m. $\mathcal{C}_2$ runs the appropriate signature generation algorithm and sends the signature to $\mathcal{A}_2$.

Forgery: $\mathcal{A}_2$ outputs a set of l devices with identity $\{I^*_{i,j}\}$, connected to the same gateway with identity I^*_i, with the set of their corresponding public keys $\{pk^*_{i,j}\}$, a set of messages $\{m^*_j\}$, and an aggregate signature σ^*_i.

The adversary $\mathcal{A}_2$ wins the game if:

- σ^*_i is a valid aggregate signature.
- Partial signing key queries have never been made for at least one identity in the set $\{I^*_{i,j}\}$ (i.e. there is at least one uncorrupted identity for devices). Secret queries have never been made for at least one identity in the set $\{I^*_{i,j}\}$ and for I^*_i (i.e. there is at least one uncorrupted identity for each entity type). W.l.o.g., let us assume that such identities are I^*_i and $I^*_{i,1}$ (i.e. $j = 1$).
- Signature queries have never been made on $(I^*_i, I^*_{i,1}, m^*_1)$.

2.5 Going further

We choose to separate KeyGen_{gat} from $\mathsf{SecretGen}_{gat}$ for the gateway, and $\mathsf{PartKeyGen}_{dev}$ from $\mathsf{SecretGen}_{dev}$ and KeyGen_{dev} for the device. Doing so, we explicitly define the steps taken by the KGC (resp. the gateway) from the gateway (resp. the device). Even if all those algorithms aim for generating keys, they are run by different entities. Separating algorithms allows to emphasize such differences.

We assume that devices can sign at the same time or sequentially. We use indexes to simplify the reading of the scheme and to identify the devices and their signatures, but do not impose any strict order among devices. When such an order is not enforced, then a time window is determined for each round, where signatures are generated and collected as long as the round is not over.

Let us now suppose that a device missed a round by not sending its signature on time. Consequently, the algorithms Aggregate and VerifyAgg are simply run without including any inputs from this missing device. Nevertheless, tracking missing devices may be beneficial if happening at multiple rounds. Extra steps may be taken to ensure that those devices are still functional. We let such features as future work.

3 OASIS Instantiation

The 2-IBAS scheme [Gritti et al.(2018b)] extends the 2-level Identity-Based Multi-Signature scheme [Gritti et al.(2018a)] by using the Gentry-Ramzan technique [Gentry and Ramzan(2006)] to allow devices to sign personal messages that are all distinct instead of signing pre-selected common messages. Informally, entities first sign a common dummy message, that is the round counter cnt, as in [Gritti et al.(2018a)], and then embed their personal message into this signatures.

Following techniques from [Al-Riyami and Paterson(2003)], OASIS alleviates the key escrow problem at the gateway level by letting the second element β of the KGC's master secret key msk in Gritti et al.'s scheme [Gritti et al.(2018b)] be the gateway's secret value β_i. This implies that the element h_2 of the public parameters in [Gritti et al.(2018b)] is actually the gateway's public key pk_i. OASIS prevents the key escrow problem at devices' level by letting each device complete its partial signing key $psk_{i,j}$ generated by the KGC with an additional random secret value $x_{i,j}$ to obtain the secret key $sk_{i,j}$, and compute the corresponding public key from that secret value.

3.1 Background

Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p according to the security parameter λ. Let g be a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map such that: (1) *Bilinearity:* $\forall u, v \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$; (2) *Non-degeneracy:* $e(g, g) \neq 1_{\mathbb{G}_T}$; (3) *Symmetry:* $\forall a, b \in \mathbb{Z}_p$, $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$. Finally, $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G} \times \mathbb{G}$ and the bilinear map e are both efficiently computable.

Computational Diffie-Hellman Assumption

We define the Computational Diffie-Hellman (CDH) problem as follows. Let $\mathbb{G}$ be a group of prime order p according to the security parameter λ. Let $a, b \in \mathbb{Z}_p$ and g be a generator of $\mathbb{G}$. The problem is: Given a CDH tuple (g, g^a, g^b), it remains hard to compute $g^{ab} \in \mathbb{G}$. The CDH assumption holds if no probabilistic polynomial-time adversary $\mathcal{A}$ has non-negligible advantage in solving the CDH problem.

3.2 Construction

To simplify the presentation of our construction, we only consider the case where l devices are connected to one gateway.

• $\mathsf{Setup}(\lambda) \rightarrow (params, msk)$. Given the security parameter λ, let $\mathbb{G}, \mathbb{G}_T$ be two cyclic multiplicative groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map. The KGC randomly chooses $\alpha \in \mathbb{Z}_p^*$ and compute $h = g^\alpha$. Let $H_1, H_2, H_3 : \{0,1\}^* \rightarrow \mathbb{G}$ and $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be four cryptographic hash functions seen as random oracles. Finally, the KGC sets the public parameters as $params = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, H_1, H_2, H_3, H_4)$ and the master secret key as $msk = \alpha$.

• $\mathsf{KeyGen}_{gat}(params, msk, I_i) \rightarrow sk_i$. The KGC computes $g_i = H_1(I_i)$ and sets the secret key sk_i of the gateway with identity I_i as $sk_i = g_i^\alpha$.

• $\mathsf{SecretGen}_{gat}(params) \rightarrow \beta_i$. The gateway picks at random $\beta_i \in \mathbb{Z}_p^*$ and sets it as its secret value.

• $\mathsf{PubKeyGen}_{gat}(params, \beta_i) \rightarrow pk_i$. The gateway computes $pk_i = g^{\beta_i}$ and sets it as its public key.

• $\mathsf{PartKeyGen}_{dev}(params, sk_i, \beta_i, I_{i,j}) \rightarrow psk_{i,j}$. Given the secret key sk_i and secret value β_i, the gateway generates the partial signing key of the device with identity $I_{i,j}$ as follows. It first computes $g_{j,0} = H_2(I_{i,j}, 0)$, $g_{j,1} = H_2(I_{i,j}, 1)$, $D_{i,j}^{(1)} = sk_i \cdot g_{j,0}^{\beta_i}$ and $D_{i,j}^{(2)} = sk_i \cdot g_{j,1}^{\beta_i}$. It then sets the partial signing key as $psk_{i,j} = (D_{i,j}^{(1)}, D_{i,j}^{(2)})$.

• $\mathsf{SecretGen}_{dev}(params) \rightarrow x_{i,j}$. The device picks at random $x_{i,j} \in \mathbb{Z}_p^*$ and sets it as its secret value.

• $\mathsf{KeyGen}_{dev}(params, psk_{i,j}, x_{i,j}) \rightarrow sk_{i,j}$. Given its partial signing key $psk_{i,j}$ and secret value $x_{i,j}$, the device computes $E_{i,j}^{(1)} = (D_{i,j}^{(1)})^{x_{i,j}}$ and $E_{i,j}^{(2)} = (D_{i,j}^{(2)})^{x_{i,j}}$ and sets the signing key as $sk_{i,j} = (E_{i,j}^{(1)}, E_{i,j}^{(2)})$.

• $\mathsf{PubKeyGen}_{dev}(params, x_{i,j}, pk_i) \rightarrow pk_{i,j}$. Given its secret value $x_{i,j}$ and the gateway's public key pk_i, the device computes $F_{i,j}^{(1)} = h^{x_{i,j}}$ and $F_{i,j}^{(2)} = pk_i^{x_{i,j}}$ and sets the public key as $pk_{i,j} = (F_{i,j}^{(1)}, F_{i,j}^{(2)})$.

• $\mathsf{Sign}(params, sk_{i,j}, m_j, cnt) \rightarrow \sigma_{i,j}$. Let the round counter cnt be a fresh string for that round. The device parses its signing key $sk_{i,j}$ as $E_{i,j}^{(1)}$ and $E_{i,j}^{(2)}$. It then randomly chooses $t_j \in \mathbb{Z}_p^*$ and computes $g_{cnt} = H_3(cnt)$, $a_j = H_4(m_j, I_{i,j}, cnt)$ and the elements $B_{i,j}^{(1)} = g_{cnt}^{t_j} \cdot E_{i,j}^{(1)} \cdot (E_{i,j}^{(2)})^{a_j}$ and $B_{i,j}^{(2)} = g^{t_j}$. The device sets the signature as $\sigma_{i,j} = (B_{i,j}^{(1)}, B_{i,j}^{(2)}, cnt)$.

• $\mathsf{Aggregate}(params, \{\sigma_{i,j}\}_{j \in [1,l]}) \rightarrow \sigma_i$. Given the (optionally verified) signatures $\sigma_{i,j} = (B_{i,j}^{(1)}, B_{i,j}^{(2)}, cnt)$, for $j \in [1, l]$, with the same counter cnt from the l devices connected to the gateway, the latter generates the aggregated elements $S_i^{(1)} = \prod_{j=1}^{l} B_{i,j}^{(1)}$ and $S_i^{(2)} = \prod_{j=1}^{l} B_{i,j}^{(2)}$. It sets the aggregate signature as $\sigma_i = (S_i^{(1)}, S_i^{(2)}, cnt)$.

• $\mathsf{Verify}(params, I_i, I_{i,j}, \sigma_{i,j}, m_j, pk_{i,j}) \rightarrow \{\text{"Accept"}, \text{"Reject"}\}$. Given the identity I_i of the gateway and the identity $I_{i,j}$ of a device connected to the gateway, the device's personal message m_j, and the corresponding signature $\sigma_{i,j} = (B_{i,j}^{(1)}, B_{i,j}^{(2)}, cnt)$, the gateway checks whether the following equation holds:

$$e(B_{i,j}^{(1)}, g) = e(H_3(cnt), B_{i,j}^{(2)}) \cdot e(H_1(I_i), (F_{i,j}^{(1)})^{1+a_j}) \cdot e(H_2(I_{i,j}, 0) \cdot H_2(I_{i,j}, 1)^{a_j}, F_{i,j}^{(2)})$$

where $a_j = H_4(m_j, I_{i,j}, cnt)$. If the above equation holds, then the gateway outputs "Accept"; otherwise, it outputs "Reject".

• $\mathsf{VerifyAgg}(params, I_i, \{I_{i,j}\}_{j\in[1,l]}, \sigma_{i,j}, \{m_j\}_{j\in[1,l]}, \{pk_{i,j}\}_{j\in[1,l]}) \rightarrow$ {"Accept", "Reject"}. Given the identity I_i of the gateway and the set of identities $\{I_{i,j}\}_{j\in[1,l]}$ of the l devices connected to the gateway, the set of their messages $\{m_j\}_{j\in[1,l]}$, and the corresponding aggregate signature $\sigma_i = (S_i^{(1)}, S_i^{(2)}, cnt)$, the verifier checks whether the following equation holds:

$$e(S_i^{(1)}, g) = e(H_3(cnt), S_i^{(2)}) \cdot \prod_{j=1}^{l} (e(H_1(I_i), (F_{i,j}^{(1)})^{1+a_j}) \cdot e(H_2(I_{i,j}, 0) \cdot H_2(I_{i,j}, 1)^{a_j}, F_{i,j}^{(2)}))$$

where $a_j = H_4(m_j, I_{i,j}, cnt)$. If the above equation holds, then the verifier outputs "Accept"; otherwise, it outputs "Reject".

3.3 Security Proofs

Correctness

Due to the page limit, we only show that the equation from VerifyAgg holds. Showing that the equation from Verify holds works similarly. Let l be the number of devices connected to the gateway. Let I_i be the identity of the gateway and $\{I_{i,j}\}_{j\in[1,l]}$ be the set of identities of the devices. Let $\sigma_i = (S_i^{(1)}, S_i^{(2)}, cnt)$ be the aggregate signature and $\{m_j\}_{j\in[1,l]}$ be the set of the devices' messages. Let $a_j = H_4(m_j, I_{i,j}, cnt)$ with cnt being the fresh counter for that round of signatures.

$$
\begin{aligned}
e(S_i^{(1)}, g) &= e(\prod_{j=1}^{l} B_{i,j}^{(1)}, g) \\
&= e(g_{cnt}^{\sum_{j=1}^{l} t_j}, g) \cdot e(g_i^{\alpha \sum_{j=1}^{l} x_{i,j}}, g) \cdot e(\prod_{j=1}^{l} g_{j,0}^{\beta_i x_{i,j}}, g) \\
&\quad \cdot e(g_i^{\alpha \sum_{j=1}^{l} a_j x_{i,j}}, g) \cdot e(\prod_{j=1}^{l} g_{j,1}^{\beta_i a_j x_{i,j}}, g) \\
&= e(g_{cnt}, S_i^{(2)}) \cdot e(g_i, \prod_{j=1}^{l} h^{x_{i,j}}) \\
&\quad \cdot e(g_i, \prod_{j=1}^{l} (h^{x_{i,j}})^{a_j}) \cdot \prod_{j=1}^{l} \left(e(g_{j,0}, pk_i^{x_{i,j}}) \cdot e(g_{j,1}^{a_j}, pk_i^{x_{i,j}}) \right) \\
&= e(H_3(cnt), S_i^{(2)}) \cdot \prod_{j=1}^{l} (e(H_1(I_i), (F_{i,j}^{(1)})^{1+a_j}) \\
&\quad \cdot e(H_2(I_{i,j}, 0) \cdot H_2(I_{i,j}, 1)^{a_j}, F_{i,j}^{(2)}))
\end{aligned}
$$

Game I: Overview of the Security Proof.

Here, we only give an overview of the proof of Game I. We show that OASIS is secure against Type-I adversaries in the random oracle model, as long as the CDH problem is hard. We let the reader refer to [Al-Riyami and Paterson(2003), Gritti et al.(2018b)] for more details. The Type-I adversary $\mathcal{A}_1$ wishes to break the security of OASIS in the random oracle model. The challenger $\mathcal{C}_1$ attempts to solve the CDH problem by interacting with $\mathcal{A}_1$. A CDH tuple (g, g^a, g^b) is given to $\mathcal{C}_1$. The hash functions H_1, H_2, H_3 and H_4 are controlled by $\mathcal{C}_1$, by managing their associated lists. W.l.o.g., we assume that there is only one sub-network, hence only one gateway with identity I_i^*, such that l devices are connected to it. We reduce the security of OASIS to that of 2-IBAS [Gritti et al.(2018b)] in which the adversary can modify the public key presented by $\mathcal{C}_1$. Such a reduction uses a specific knowledge extractor $\mathcal{KE}$ algorithm to manage signature queries. The knowledge extraction algorithm $\mathcal{KE}$ has access to the lists of H_3 and H_4. Then, we reduce to that of the difficulty in solving the CDH problem.

Game II: Sketch of the Security Proof.

We only sketch the proof of Game II. We show that OASIS is secure against Type-II adversaries in the random oracle model, as long as the CDH problem is hard. We provide some intuition on a challenger $\mathcal{C}_2$ being successful in solving the CDH problem while interacting with a Type-II adversary $\mathcal{A}_2$. We let the reader refer to [Gentry and Ramzan(2006), Gritti et al.(2018a), Gritti et al.(2018b)] for more details. The Type-II adversary $\mathcal{A}_2$ wishes to break the security of OASIS in the random oracle model. The challenger $\mathcal{C}_2$ attempts to solve the CDH problem by interacting with $\mathcal{A}_2$. A CDH tuple (g, g^a, g^b)

is given to $\mathcal{C}_2$. The hash functions H_1, H_2, H_3 and H_4 are controlled by $\mathcal{C}_2$. W.l.o.g., we assume that there is only one sub-network, hence only one gateway with identity I_i^*, such that l devices are connected to it. The idea of the proof is to let the public key of the gateway with identity I_i^* be $pk_i^* = g^b$. Implicitly, β_i^* is equal to b. Therefore, we let the adversary $\mathcal{A}_2$ submit public key and secret queries on identities of devices connected to the gateway with identity I_i^*. We now describe how the challenger $\mathcal{C}_2$ replies to queries involving hash computations.

H_1 **queries:** To answer, $\mathcal{C}_2$ randomly chooses an element ν_i in $\mathbb{Z}_p$ and computes $g_i = g^{\nu_i}$. We recall that the adversary $\mathcal{A}_2$ has access to the master secret key $msk = \alpha$, and thus can compute g_i^{α}.

H_2 **queries:** To answer, $\mathcal{C}_2$ picks at random $\mu_{j,0}, \mu_{j,1} \in \mathbb{Z}_p$ and defines $g_{j,0}^{\beta_i^*}$ as $g^{b\mu_{j,0}}$ and $g_{j,1}^{\beta_i^*}$ as $g^{b\mu_{j,1}}$. Nevertheless, $\mathcal{C}_2$ sometimes computes $g_{j,0} = g^{\mu_{j,0}} \cdot (g^a)^{\mu'_{j,0}}$ and $g_{j,1} = g^{\mu_{j,1}} \cdot (g^a)^{\mu'_{j,1}}$ for some elements $\mu'_{j,0}, \mu'_{j,1} \in \mathbb{Z}_p$. Thus, in such a situation, $\mathcal{C}_2$ is not able to answer to a partial signing key generation query on identity $I_{i,j}$. However, in the case of this identity $I_{i,j}$ being the target choice of $\mathcal{A}_2$, the forgery of the latter could help $\mathcal{C}_2$ solve the CDH problem.

H_3 **queries:** To answer, $\mathcal{C}_2$ computes $g_{cnt} = (g^a)^{d_{cnt}}$ for a known random exponent $d_{cnt} \in \mathbb{Z}_p$ most of the time. Nevertheless, it sometimes computes $g_{cnt} = g^{c_{cnt}}$ for another known random exponent $c_{cnt} \in \mathbb{Z}_p$.

H_4 **queries:** To answer, $\mathcal{B}$ randomly chooses an element ξ_j and computes $a_j = g^{\xi_j}$ if it knows d_{cnt}. Otherwise, it calculates $a_j = g^{\xi}$ such that the exponent ξ is a unique value that helps cancel out the multiple of g^{ab} in $B_{i,j}^{(1)}$.

$\mathcal{C}_2$ is able to reply to a signature query on identity $I_{i,j}$, counter cnt and message m_j by controlling the H_2, H_3 and H_4 oracles, although it cannot get the signing key linked to the identity $I_{i,j}$. There are two cases: (1) $\mathcal{C}_2$ knows d_{cnt}, from $g_{cnt} = (g^a)^{d_{cnt}}$. Then, $\mathcal{C}_2$ computes the value of the exponent t' such that the value $g_{cnt}^{bt'}$ deletes the multiple of g^{ab} that comes in the other terms of the signing element $B_{i,j}^{(1)}$. It finally sets $B_{i,j}^{(2)} = (g^b)^{t'}$; (2) $\mathcal{C}_2$ does not know d_{cnt}, from $g_{cnt} = (g^a)^{d_{cnt}}$. However, it can sometimes fix the exponent $a_j = H_4(m_j, I_{i,j}, cnt)$ to be the unique value in $\mathbb{Z}_p^*$ such that the multiples of g^{ab} cancel out in the signing element $B_{i,j}^{(1)}$. Hence, $\mathcal{C}_2$ is able to generate a valid signature. If the unique value a_j is divulged for the identity $I_{i,j}$, then $\mathcal{C}_2$ is allowed to re-use this trick later. Suppose now that $\mathcal{C}_2$ does not abort, $\mathcal{A}_2$ gives a forgery on identity $I_{i,j}^*$, message m_j^* and counter cnt for which the exponents $\mu_{j,0}$ and $\mu_{j,1}$, from $g_{j,0}^{\beta_i^*} = g^{b\mu_{j,0}}$ and $g_{j,1}^{\beta_i^*} = g^{b\mu_{j,1}}$, are not known, and the exponent a_j is not determined regarding the aforementioned trick. Then, $\mathcal{C}_2$ obtains the value of g^{ab} with high probability given $\mathcal{A}_2$'s forgery.

4 OASIS Evaluation

4.1 Comparison with 2-IBAS

The 2-IBAS scheme is the closest to OASIS in terms of computational benchmark. The former was analysed and claimed as deployable in IoT [Gritti et al.(2018b)]. Let us consider one gateway and l devices connected to it. Compared to 2-IBAS, there are $l+1$ extra random secret values, one extra exponentiation for the gate-

way's public key, $2l$ extra exponentiations for the devices' signing keys and $2l$ extra exponentiations for the devices' public keys in OASIS. Devices' signature generation and verification. along with signature aggregation, incur the same amount of operations in OASIS and 2-IBAS. The overall operational process of the aggregate signature verification is similar in both schemes; however the number of operations differs in OASIS and 2-IBAS. We detail those changes in Table 2.

Type of operation	Number in 2-IBAS	Number in OASIS
Pairing	4	$2 + 2l$
Multiplication in $\mathbb{G}_T$	2	$l + 1$
Exponentiation	$2 + l$	$2l$
Addition in $\mathbb{Z}_p$	l	l
Multiplication in $\mathbb{G}$	$l + 1$	l

Table 2: Numbers of operations for VerifyAgg in 2-IBAS and OASIS.

Running VerifyAgg is more cumbersome in OASIS than in 2-IBAS, with roughly $2l$ extra pairing computations and twice more exponentiations. However, verification is led by a powerful verifier, which is not limited in terms of computation, communication and storage, contrary to the devices within the network. Hence, it would not impact the feasibility of OASIS in IoT distributed systems.

4.2 Implementation

We have implemented our solution in a sub-network with one gateway and l devices. We chose the cryptographic library MIRACL[1], an open source SDK for elliptic curve cryptography. MIRACL enables to build security into PC but also constrained environments, such as IoT. We used a processor 2.4 GHz Intel i5 520M to run our tests. We tested our solution with $l = 10$, 50 and 100 devices to represent a wide range in one sub-network. We did not include the algorithms $\mathsf{SecretGen}_{gat}$ and $\mathsf{SecretGen}_{dev}$ since there is no costly operation. Table 3 lists the four Super-Singular Curves (SSCs) proposed by MIRACL. MIRACL enables pre-computation mechanisms to optimize costly pairing calculations.

Curves	Field	Modulus/exponent	Embedding degree	AES security
SSC 1	$\mathbb{GF}(p)$	512-bit modulus	2	80 bits
SSC 2	$\mathbb{GF}(p)$	1536-bit modulus	2	128 bits
SSC 3	$\mathbb{GF}(2^m)$	$m = 379$	4	80 bits
SSC 4	$\mathbb{GF}(2^m)$	$m = 1223$	4	128 bits

Table 3: Super-singular curves provided by MIRACL.

Timings, shown in Table 4, were collected per algorithm, based on exponentiations and multiplications in $\mathbb{G}$, and pairings in $\mathbb{G}_T$. The algorithms Sign and

[1] https://github.com/miracl/MIRACL/tree/master

Verify were run for one device. The algorithms Aggregate and VerifyAgg were run for l devices, where $l = 10$, 50 or 100. Pairing calculation optimization is denoted as (o).

Algorithms \ Curves	SSC 1	SSC 2	SSC 3	SSC 4
Setup	1.49	0.38	13.57	2.57
(o)	0.30	–	3.01	–
KeyGen_{gat}	1.49	0.38	13.57	2.57
(o)	0.30	–	3.01	–
$\mathsf{PubKeyGen}_{gat}$	1.49	0.38	13.57	2.57
(o)	0.30	–	3.01	–
$\mathsf{PartKeyGen}_{dev}$	4.47	1.14	40.71	7.71
(o)	0.90	–	9.03	–
KeyGen_{dev}	2.98	0.76	27.14	5.14
(o)	0.60	–	6.02	–
$\mathsf{PubKeyGen}_{dev}$	2.98	0.76	27.14	5.14
(o)	0.60	–	6.02	–
Sign	7.45	1.90	67.85	12.85
(o)	1.50	–	15.05	–
Aggregate for $l = 10$	11.92	3.04	108.56	20.56
(o)	2.40	–	24.08	–
Aggregate for $l = 50$	71.52	18.24	651.36	123.36
(o)	14.4	–	144.48	–
Aggregate for $l = 100$	146.02	37.24	1329.86	251.86
(o)	29.4	–	294.98	–
Verify	13.19	4.97	130.49	77.54
(o)	4.05	–	47.17	–
VerifyAgg for $l = 10$	81.23	29.99	749.15	444.20
(o)	20.25	–	196.93	–
Aggregate for $l = 50$	383.63	141.19	3498.75	2073.80
(o)	92.25	–	862.53	–
VerifyAgg for $l = 100$	761.63	280.19	6935.75	4110.80
(o)	182.25	–	1694.53	–

Table 4: Timings in milliseconds.

Selecting the curve SSC 3 implies bigger times for all algorithms. However, with pairing calculation optimization, then timings are similar to SSC 4, for which the optimization is not supported. Moreover, SSC 1 enables the system to run 10x faster than with SSC 3, for the same security level. Similarly, the system runs almost 15x slower with SSC 4 compared to SSC 2. Hence, without optimization and a higher security level, the best results come with SSC 2. With optimization and a lower security level, the best results come with SSC 1.

Algorithms for key generation operate fast when run independently. However, when considering those six algorithms and the two additional ones for secret generation as one unique algorithm, key generation noticeably takes a longer time. A distributed network includes multiple sub-networks, so multiple gateways and devices. Hence, timings for key generation would depend on the total number of entities. However, such remarks could be mitigated since key

generation is handled by powerful entities (KGC and gateways) and remains occasional (static keys).

Signing remains quicker than verifying, since the former has no pairing calculation, which is the most expensive operations. The algorithm Aggregate runs faster than VerifyAgg for the same reasons. Execution times for Aggregate and VerifyAgg are roughly linear in the number l of signers in a sub-network, since their number of operations is $O(l)$. Pairing pre-computations, available for SSC 1 and SSC 3, are beneficial; however, at the cost of a lower security level (80 bits). We recall that the verifier is powerful with no resource constraints and checks much less signatures compared to the total number of devices in the network thanks to the aggregation mechanism, hence yielding pairings acceptable.

As for most of IoT networks, execution times depend on the total number of participating entities. Nevertheless, our distributed architecture enables to keep a number of devices relatively low in each sub-network. For instance, 20 sensors per human body would be installed in a HWMSN [Ragesh and Baskaran(2012)]. Therefore, OASIS could be deployed in an IoT distributed environment; especially when implemented with SSC 1/(o) for 80-bit security and with SSC 2 for 128-bit security.

5 Conclusion and Future Work

We presented OASIS, a new CLAS scheme for IoT with an organizational architecture based on 2 levels, mitigating the complex PKI certification management. To alleviate the key escrow problem succinct to identity-based schemes, OASIS enables gateways and devices to create their own secret value. We gave intuitions to prove our scheme secure in the random oracle model. We also evaluated our solution to verify its suitability in IoT distributed networks.

OASIS considers a 2-level hierarchy, with a KGC, few gateways and multiple devices. Future work will consider an organizational chart with N levels to obtain a more generic solution in larger distributed networks. Moreover, we quickly discussed how devices missing a round could be handled. Future work will focus on formal mechanisms to track those missing devices.

References

[Al-Riyami and Paterson(2003)] S. S. Al-Riyami and K. G. Paterson. 2003. Certificateless public key cryptography. In *Proc. Cryptology-ASIACRYPT*, Vol. 2894. Lecture Notes Computer Science, Springer, 452–473.

[Ameen et al.(2012)] M. A. Ameen, J. Liu, and K. Kwak. 2012. Security and privacy issues in wireless sensor networks for healthcare applications. *J. Med. Syst.*. 36, 1 (2012), 93–101.

[Au et al.(2007)] M. H. Au, Y. Mu, J. Chen, D. S. Wong., J. K. Liu, and G. Yang. 2007. Malicious KGC attacks in certificateless cryptography. In *Proc. ACM Symp. Inf. Comput. Commun. Sec.* 302–311.

[Boneh et al.(2003)] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc.*

Cryptology-EUROCRYPT, Vol. 2656. Lecture Notes Computer Science, Springer, 416–432.

[Cui et al.(2018)] J. Cui, J. Zhang, H. Zhong, R. Shi, and Y. Xu. 2018. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Inf. Sci.* 451 (2018), 1–15.

[de Schatz et al.(2012)] C. H. Vallejos de Schatz, H. P. Medeiros, F. K. Schneider, and P. J. Abatti. 2012. Wireless medical sensor networks: design requirements and enabling technologies. *Telemed J. E. Health* 18, 5 (2012), 394–399.

[Gayathri et al.(2019)] N. B. Gayathri, G. Thumbur, P. R. Kumar, M. Z. U. Rahman, P. V. Reddy, and A. Lay-Ekuakille. 2019. Efficient and secure pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor networks. *IEEE Internet of Things J.* 6, 5 (2019), 9064–9075.

[Gentry and Ramzan(2006)] C. Gentry and Z. Ramzan. 2006. Identity-Based Aggregate Signatures. In *Proc. Int. Conf. Theor. Pract. PKC*. 257–273.

[Gong et al.(2007)] Z. Gong, Y. Long, X. Hong, and K. Chen. 2007. Two certificateless aggregate signatures from bilinear maps. In *Proc. ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw., Parallel/Distrib. Comput.* 188–193.

[Gritti et al.(2018a)] C. Gritti, R. Molva, and M. Önen. 2018a. Lightweight secure bootstrap and message attestation in the Internet of Things. In *Proc. ACM/SIGAPP Symp. on Appl. Comp.* 775–782.

[Gritti et al.(2018b)] C. Gritti, M. Önen, R. Molva, W. Susilo, and T. Plantard. 2018b. Device Identification and Personal Data Attestation in Networks. *J. of Wir. Mob. Net. Ubiq. Comp. Dep. App. (JoWUA)* 9 (2018), 1–25.

[He et al.(2013)] D. He, B. Huang, and J. Chen. 2013. New certificateless short signature scheme. *IET Inf. Secur.* 7, 2 (2013), 113–117.

[Huang et al.(2005)] X. Huang, W. Susilo, Y. Mu, and F. Zhang. 2005. On the security of certificateless signature schemes from Asiacrypt 2003. In *Proc. Int. Conf. Cryptol. Netw. Sec.* 13–25.

[Kamil and Ogundoyin(2019)] I. A. Kamil and S. Ogundoyin. 2019. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *J. Inf. Sec. Appl.* 44 (2019), 184–200.

[Ko et al.(2010)] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh. 2010. Wireless sensor networks for healthcare. *Proc. of the IEEE* 98, 1 (2010), 1947–1960.

[Kumar et al.(2018)] P. Kumar, S. Kumari, V. Sharma, A. K. Sangaiah, J. Wei, and X. Li. 2018. A certificateless aggregate signature scheme for healthcare wireless sensor network. *Sustain. Comput. Informatics Syst.* 18 (2018), 80–89.

[Kumar and Lee(2012)] P. Kumar and H. J. Lee. 2012. Security issues in healthcare applications using wireless medical sensor networks: A survey. *Sensors* 12, 1 (2012), 55–91.

[Liu et al.(2020)] J. Liu, L. Wang, and Y. Yu. 2020. Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks. *IEEE Internet of Things J.* 7, 6 (2020), 5256–5266.

[Ragesh and Baskaran(2012)] G. Ragesh and K. Baskaran. 2012. An overview of applications, standards and challenges in futuristic wireless body area networks. *Int. J. Comput. Sci. Issue.* 9, 2 (2012), 1694–1814. Issue 1.

[Saeed et al.(2018)] M. E. S. Saeed, Q. Y. Liu, G. Tian, B. Gao, and F. Li. 2018. Remote authentication schemes for wireless body area networks based on the Internet of Things. *IEEE Internet of Things J.* 5, 6 (2018), 4926–4944.

[Shen et al.(2018)] L. Shen, J. Ma, H. Liu, and Y. Miao. 2018. A provably secure aggregate signature scheme for healthcare wireless sensor network. *Sustain. Comput. Informat. Syst.* 18 (2018), 80–89.

[Shen et al.(2017)] L. Shen, J. Ma, H. Liu, F. Wei, and Y. Miao. 2017. A secure and efficient ID-based aggregate signature scheme for wireless sensor networks. *IEEE Internet of Things J.* 4, 2 (2017), 546–554.

[Shen et al.(2019)] L. Shen, J. Ma, Y. Miao, and H. Liu. 2019. Provably secure certificateless aggregate signature scheme with designated verifier in an improved security model. *IET Inf. Secur.* 13, 3 (2019), 167–173.

[Thumbur et al.(2021)] G. Thumbur, G. S. Rao, P. V. Reddy, N. B. Gayathri, D. V. R. K. Reddy, and M. Padmavathamma. 2021. Efficient and secure certificateless aggregate signature-based authentication scheme for vehicular Ad Hoc Networks. *IEEE Internet of Things J.* 8, 3 (2021), 1908–1920.

[Wu et al.(2018)] L. Wu, Z. Xu, D. He, and X. Wang. 2018. New certificateless aggregate signature scheme for healthcare multimedia social network on cloud environment. *Sec. Comm. Netw.* (2018), 1–13.

[Y. Zhan and Lu(2021)] B. Wang Y. Zhan and R. Lu. 2021. Cryptanalysis and improvement of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks. *IEEE Internet of Things J.* 8, 7 (2021), 5973–5984.

[Yang et al.(2021)] W. Yang, S. Wang, and Y. Mu. 2021. An enhanced certificateless aggregate signature without pairings for e-healthcare system. *IEEE Internet of Things J.* 8, 6 (2021), 5000–5008.

[Zhang and Zhang(2009)] L. Zhang and F. Zhang. 2009. A new certificateless aggregate signature scheme. *Comput. Commun.* 32, 6 (2009), 1079–1085.

[Zhang et al.(2019)] Y. Zhang, J. Shu, X. Liu, J. Li, and D. Zheng. 2019. Comments on a large-scale concurrent data anonymous batch verification scheme for mobile healthcare crowd sensing. *IEEE Internet of Things J.* 6, 1 (2019), 1287–1290.