Efficiently Validating Aggregated IoT Data Integrity

Nesrine Kaâniche*
SAMOVAR, Telecom SudParis
CNRS, University Paris-Saclay, France
Email: nesrine.kaaniche@telecom-sudparis.eu

Eunjin (EJ) Jung
Department of Computer Science
University of San Francisco
Email: ejung@cs.usfca.edu

Ashish Gehani
Computer Science Laboratory
SRI International
Email: ashish.gehani@sri.com

***Abstract*—We address the problem of validating the integrity of data aggregated in applications that run on the Internet of Things (IoT). When data from multiple sources is sent to a single receiver, the receiver needs to efficiently verify the authenticity and integrity of data. For example, sensors from multiple IoT devices would send the measurements to a gateway for data aggregation and/or processing. The gateway needs to verify the authenticity and integrity of the readings from sensors. To address this need, we propose a lightweight homomorphic signature scheme that supports the execution of aggregation operations.**

Our approach collects digital signatures from multiple sources and provides an efficient aggregated signature verification mechanism for the end receiver. The aggregation and verification mechanism preserves the privacy of intermediate nodes. We discuss how our homomorphic signature scheme resists forgery and replay attacks.

I. Introduction

During the last decade, the Internet of Things (IoT) penetrated our lives in many ways – most recently in smart monitoring and control applications at home and workplaces. IoT devices are smart, and often resource-constrained devices, and they interact in a collaborative way or a semi-centric fashion under different processing and communication architectures and technologies in order to fulfill a common goal [1]. For example, a set of resource-constrained IoT devices can collect environmental information regarding a targeted area and collaboratively transmit data back to a base station or a central aggregator to monitor air quality, earthquake activities, or traffic patterns.

In many cases, users of IoT applications are only interested in aggregated results after in-network processing, rather than detailed readings from individual nodes. To ensure the authenticity and integrity of the data in the aggregated results, we need to provide a mechanism for the aggregator nodes to create a digital signature that matches the aggregated data, and for the end receiver to verify this signature.

While there are known cryptographic mechanisms that can provide data confidentiality, data integrity, and data/device authentication in a traditional computing environment of personal computers and servers, these may require heavy computation and communication costs. Instead of relying on dedicated resourceful devices, we aim to take advantage of large and distributed computation and storage capacities of the different devices by requiring each device to provide signed information for collected data. Each device aggregates the data it has received so far and is responsible for creating matching signature for the aggregated data.

Aggregation has been often presented as a solution to take advantage of the local computing and storage capabilities in order to remove redundant information within network flows. Many aggregation mechanisms have been proposed in the literature [2]–[4]. The use of such approaches considerably distributes the heavy computation burden among different hosts, thereby reducing the resource consumption of the end receiver. However, aggregation affects the security properties of protection systems. For instance, if some information of a signed data flow is dropped to reduce redundancy, the end receiver may not be able to correctly verify the authenticity of the signed stream.

Contributions – Our aggregation signature scheme allows the resource saving of aggregation and also allows the nodes to create the corresponding signatures of any subset of aggregate data for authenticity without knowing the private key materials of the original senders of the aggregated data. We propose a new homomorphic signature scheme based on set operations. A set-homomorphic signature scheme allows signing sets of data such that anyone can derive both a signature on the union of two signed sets and a signature of any subsets of a signed set [5]. We consider all data sent from each IoT device as an element of a set. This enables us to design an efficient signature scheme that allows the aggregating entity to do the following:

1) detect forged signatures before the data and signatures reach the end receiver.
2) generate a proper signature without private keys used to derive each single signature.
3) apply set operations on signed data flows, such as union, subset, and set difference.

Paper Organization – The remainder of this paper is organized as follows. First, we detail the requirements needed for our signature scheme and review the related approaches presented in the literature in Section II. Then, we introduce our homomorphic signature scheme in Section III and present a concrete construction in Section IV. Finally, we give a detailed analysis in Section V, before concluding in Section VI.

* while Dr. Kaâniche was visiting SRI International

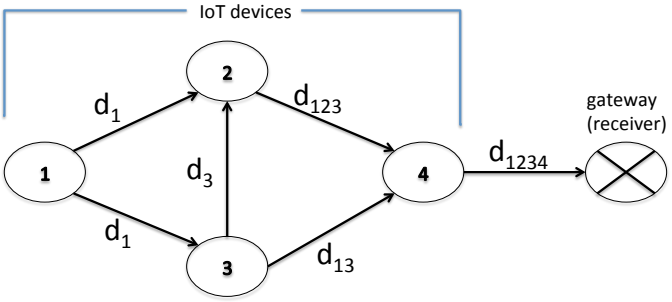

Fig. 1. An Example System

II. Digital Signature Schemes for Distributed Environments

IoT devices involved in distributed environments provide ubiquitous and pervasive computing capabilities. However, these systems are often deployed in environments where collecting hosts are open to various attacks due to untrusted environments. We introduce our network model in Section II-A and present security and privacy requirements in Section II-B.

A. Network Model

Figure II-A shows an example system with IoT devices and a gateway. Each IoT device periodically broadcasts the data it received from other IoT nodes and its own data (e.g. readings from sensors.) d_i stands for data from node i, and d_{ij} stands for the aggregated data from nodes i and j. The aggregated data (d_{ij} can be in any form, such as concatenation ($d_i||d_j$) average ($(d_i + d_j)/2$), or max (max(d_i, d_j)). Note that node 3 aggregates all it has received (d_1) and its own d_3 into d_{13}, while node 4 aggregates d_{123} and d_{13} into d_{1234}, without repeating d_1 and d_3. In other words, the aggregation step removes the redundancy in data.

Our aggregated signature scheme enables node 3 to create a signature for d_{13} using the signature for d_1 and its own private key without knowing node 1's private key. Similarly, node 4 can create a signature for d_{1234} using the signatures of d_{123} and d_{13} and its own private key without knowing the private key of any other nodes.

B. Requirement Analysis

We suppose the aggregation process can be performed by any participating node, and it may result in the removal of some information to reduce the redundancy between the collected data stream transmitted from different nodes to minimize communication latency.

Considering that collected data can be output by different nodes, our signature scheme has to fulfill the following requirements:

- *Unforgeability* – even if an adversary can capture signatures of different contents, it should be unable to generate a new signature on new data contents.
- *Anti-replay* – for two communication sessions, even if the content of the collected packets is the same, the generated signatures should be different. This prevents any adversary to re-inject responses previously signed by a specific host.
- *Privacy* – an adversary must not be able to gain any knowledge about the origin and content of the removed parts of the signed data flow without having access to them or to link each part of an aggregate signature to its related generator.
- *Support of multiple signers* – the computation of the aggregated signature should be possible even if multiple private keys are used.
- *Low processing and communication overhead* – the proposed security mechanism should provide low processing and communication complexities due to resource-constrained devices.

C. Literature Review

Various types of homomorphic signatures have been presented in the literature [2], such as sanitizable signatures [6], set-based signatures [5], redactable signatures [7], linearly homomorphic signatures [8], and verifiably encrypted signatures [3]. The constructions in [9], [10] propose sanitizable and redactable signatures, respectively. They enable an authorized user to bring modifications to a signed message. That is, given a signature on a message, a signature on subsets of this message can be generated by the authorized entity. However, these techniques generate long signatures that require substantial computing resources and introduce communication overhead. The mechanisms in [3], [11], [12] mechanisms are limited with regard to the supported homomorphic operations, allowing only polynomial, transitive, multiplication operations, respectively.

III. System Overview

Our objective is to develop a new distributed aggregate signature scheme. We discuss the basic properties that should be fulfilled by such a scheme in section III-A. Then, we introduce our system model in Section III-B and our threat model in Section III-C.

A. Basic Properties

Definition 3.1: **Set-based Signature Scheme**
Let $\mathcal{M}$ be a message space, Σ be a signature space, $\mathcal{K}_{pr}$ a private key space, and $\mathcal{K}_{pub}$ a public key space. A set-based signature scheme consists of the following three algorithms:

- $\texttt{gen} : 1^{\xi} \longrightarrow \mathcal{K}_{pr} \times \mathcal{K}_{pub}$: this key generation algorithm takes as input the security parameter ξ and outputs the public and private keys.
- $\texttt{sig} : \mathcal{K}_{pr} \times 2^{\mathcal{M}} \longrightarrow \Sigma \times 2^{\mathcal{M}}$: the signature generation algorithm takes as input the private key of the signing host and the set of messages. It outputs the related signature. Note that $2^{\mathcal{M}}$ is the power set of $\mathcal{M}$.
- $\texttt{vrf} : \mathcal{K}_{pub} \times 2^{\mathcal{M}} \times \Sigma \longrightarrow \{0, 1\}$: the signature verification algorithm takes as input the signature and the public key related to the signing key, and it outputs 1; i.e *accept*; or 0; i.e *reject*.

The main difference between traditional signature schemes is that the sig algorithm operates on sets of messages in $2^{\mathcal{M}}$ instead of operating on individual messages in $\mathcal{M}$.

Definition 3.2: **Set-homomorphic Signature Scheme** A set-based signature scheme $\{\texttt{gen}, \texttt{sig}, \texttt{vrf}\}$ is called set homomorphic if these three operations exist $\circ : \mathcal{K}_{pr} \times \mathcal{K}_{pr} \longrightarrow \mathcal{K}_{pr}$, $\Diamond : \mathcal{K}_{pub} \times \mathcal{K}_{pub} \longrightarrow \mathcal{K}_{pub}$ and $\nabla : \Sigma \times \Sigma \longrightarrow \Sigma$ that satisfy the following two properties for a set operation $\star$ and for any sets S_i and S_j and for any sk_i and sk_j in $\mathcal{K}_{pr}$ and any pk_i and pk_j in $\mathcal{K}_{pub}$:

- **Homomorphism –**

$$\texttt{sig}(sk_i \circ sk_j, S_i \star S_j) = \texttt{sig}(sk_i, S_i) \nabla \texttt{sig}(sk_j, S_j) \quad (1)$$

- **Correctness –**

$$\texttt{vrf}(pk_i \Diamond pk_j, \texttt{sig}(sk_i \circ sk_j, S_i \star S_j)) =$$
$$\texttt{vrf}(pk_i, \texttt{sig}(sk_i, S_i)) \wedge \texttt{vrf}(pk_j, \texttt{sig}(sk_j, S_j)) \quad (2)$$

We note the $\wedge$ symbol is the logical AND operator.

Therefore, a Set-Homomorphic Signature Scheme (SHSS) consists of four algorithms and four functions such that:

$$\texttt{SHSS} = \{\{\texttt{gen}, \texttt{sig}, \texttt{vrf}, \texttt{agg}\}, \{\circ, \Diamond, \nabla, \star\}\}, \quad (3)$$

where the agg algorithm is defined as $\texttt{agg} : \Sigma \times \Sigma \longrightarrow \Sigma \times \mathcal{A}$ (cf; Equation 4). This algorithm returns an aggregate signature with a set of parameters in $\mathcal{A}$. We have to note $\mathcal{A}$ is the set of parameters required to calculate the $\Diamond$ function.

$$\texttt{agg}(\texttt{sig}(sk_i, S_i), \texttt{sig}(sk_j, S_j)) = \texttt{sig}(sk_i, S_i) \Diamond \texttt{sig}(sk_j, S_j) \quad (4)$$

B. System Model

Our proposal is composed of six randomized algorithms defined as follows:

$\texttt{stp} : 1^{\xi} \longrightarrow \mathcal{P}$: this algorithm is run by a trusted entity. It takes as input the security parameter ξ, and it outputs a set of public parameters $\mathcal{P}$. We suppose that $\mathcal{P}$ is an auxiliary input of all the following algorithms.

$\texttt{gen}_u : \mathcal{P} \longrightarrow \{sk_u, pk_u\}$: this algorithm is executed by a trusted entity in order to generate a pair of keys for each requesting host H_u. It takes as input the public parameters $\mathcal{P}$, and it outputs the secret key sk_u and the public key pk_u of H_u.

$\texttt{sig} : sk_u \times \mathcal{S} \longrightarrow \sigma$: this algorithm is performed by any involved host H_u. It takes as input the secret key of the signing host sk_u and the data set $\mathcal{S}$. It outputs a signature σ.

$\texttt{vrf} : pk_u \times \sigma \longrightarrow \{0, 1\}$: this algorithm is run by a verifying host. It takes as input the public key of the signing entity pk_u and the signed set $\mathcal{S}$. It outputs either 1; i.e *accept*; or 0; i.e. *reject*.

$\texttt{agg} : \{\sigma_i\}_{i \in [1,n]} \times \mathcal{A} \longrightarrow \Sigma$: this algorithm is performed by an aggregating node. It takes as input a set of n signatures and a set of parameters $\mathcal{A}$ required for the calculation of the resulting signature. It outputs an aggregated signature Σ.

$\texttt{vrfagg} : \Sigma \times \{pk_i\}_{i \in [1,n]} \times \mathcal{A} \longrightarrow \{0, 1\}$: this algorithm is executed by any verifying entity. It takes as input an aggregated signature Σ, the set of public keys of the involved hosts, and the set of aggregation parameters $\mathcal{A}$. It outputs either 1 if the signature is accepted, or 0 if rejected.

C. Security Model

To design the most suitable security solutions for distributed environments, we have to consider realistic threat models. For example, an external malicious adversary may intend to inject false information as a legitimate user, or to gain extra knowledge about collected signed data. Thus, this attacker is considered against the unforgeability, anti-replay, and the privacy-preserving properties, as defined in Section II-B.

IV. Construction

In this section, we first introduce some mathematical notations and present our cryptographic assumptions in Section IV-A. Then, we detail the different algorithms of our construction in Section IV-B.

A. Mathematical Background

In this section, we first define our collision-resistant hash function, in Subsection IV-A1. Then, we introduce bilinear maps in Section IV-A2 and detail our cryptographic assumptions in Section IV-A3.

1) Hash functions: h is a collision-resistant hash function. $h : \mathcal{M} \rightarrow \mathcal{P}$, where $\mathcal{M}$ is the message space and $\mathcal{P}$ is a set of prime numbers such that $|\mathcal{M}| \leq |\mathcal{P}|$. For all $m_i, m_j \in \mathcal{M}$, if $m_i \neq m_j$, then $h(m_i) \neq h(m_j)$ and $gcd(h(m_i), h(m_j)) = 1$.

2) Bilinear maps: Let us consider three cyclic multiplicative groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map that has to fulfill the following properties:

- *Bilinearity* – for each $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have the equality $\hat{e}({g_1}^a, {g_2}^b) = \hat{e}(g_1, g_2)^{ab}$
- *Computability* – for each $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, there exists an algorithm that efficiently calculates $\hat{e}(g_1, g_2)$
- *Non-degeneracy* – for each $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, we have $< \hat{e}(g_1, g_2) >= \mathbb{G}_T$

3) Complexity assumptions: For our construction, we consider the following complexity assumptions:

- ***Computational Diffie Hellman Assumption (CDH) –*** Let $\mathbb{G}$ be a group of a prime order p, and g is a generator of $\mathbb{G}$. The CDH problem is given the tuple of elements (g, g^a, g^b) where $\{a, b\} \xleftarrow{R} \mathbb{Z}_p$, and there is no efficient probabilistic algorithm $\mathcal{A}_{CDH}$ that computes g^{ab}.
- ***Computational co-Diffie Hellman Assumption (co-CDH) –*** Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of a prime order p, g_1, g_2 and their respective generators and $\hat{e}$ an asymmetric pairing function. The co-CDH problem in the asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \hat{e})$ is, given a tuple of elements $(g_1, g_2, {g_1}^a, {g_2}^b)$ where $\{a, b\} \xleftarrow{R} \mathbb{Z}_p$ there is no efficient probabilistic algorithm $\mathcal{A}_{co-CDH}$ that computes ${g_1}^{ab}$.

B. Concrete Construction

Our SHSS construction is composed of three phases, namely the *generation* phase, the *single signature* phase, and the *aggregation* phase.

1) Generation phase: The *generation* phase is executed only one time, and it relies on the stp and $\texttt{gen}_u$ algorithms. The stp algorithm is given by Algorithm 1. We consider the prime order p, the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and the pairing $\hat{e}$ have already been chosen and are naturally included in the parameters $\mathcal{P}$.

Algorithm 1 stp algorithm

1: select g_1 a generator of $\mathbb{G}_1$;
2: select g_2 a generator of $\mathbb{G}_2$;
3: select h a collision-resistant hash function defined in Section IV.A.1);
4: **return** $\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g_1, g_2, h)$

Then, for each host H_u, the trusted entity executes the $\texttt{gen}_u$ algorithm given by Algorithm 2.

Algorithm 2 $\texttt{gen}_u$ algorithm

1: $sk_u \xleftarrow{R} \mathbb{Z}_p$;
2: $pk_u \leftarrow {g_2}^{sk_u}$;
3: **return** (sk_u, pk_u)

2) Single signature phase: As introduced in Section III-B, the *single signature* phase is composed of two algorithms, namely sig and vrf, defined by Algorithm 3 and Algorithm 4, respectively.

Algorithm 3 sig algorithm

1: **input:** host secret key sk_u and the set $\mathcal{S} = \{m_1, m_2, \cdots, m_{n_S}\}$
2: **output:** the signature σ_S
3: $\mathcal{H}(\mathcal{S}) = 1$;
4: $n_{\mathcal{S}} \leftarrow |\mathcal{S}|$;
5: **for** $i \in [1 \dots n_{\mathcal{S}}]$ **do**
6: $\quad \mathcal{H}(\mathcal{S}) \leftarrow \mathcal{H}(\mathcal{S}) * h(m_i)$;
7: **end for**
8: $\sigma_S \leftarrow [{g_1}^{sk_u}]^{\mathcal{H}(\mathcal{S})^{-1}}$;
9: **return** σ_S

The correctness of our proposed single signature, for any set S, is easily deduced using the bilinearity property of pairing functions (cf; Equation 5) such that:

$$\hat{e}(\sigma_S, g_2) = \hat{e}([{g_1}^{sk_u}]^{\mathcal{H}(\mathcal{S})^{-1}}, g_2) = \hat{e}([g_1]^{\mathcal{H}(\mathcal{S})^{-1}}, pk_u) \quad (5)$$

This proves Equation 5 is correct if and only if the public key $pk_u = {g_2}^{sk_u}$ used by the vrf algorithm corresponds to the private key used to generate the signature σ_S.

Algorithm 4 vrf algorithm

1: **input:** host u's public key pk_u, the set $\mathcal{S}$ and the signature σ
2: **output:** boolean value $B \in \{0, 1\}$
3: $\mathcal{H}(\mathcal{S}) = 1$; $n_{\mathcal{S}} \leftarrow |\mathcal{S}|$;
4: **for** $i \in [1 \dots n_{\mathcal{S}}]$ **do**
5: $\quad \mathcal{H}(\mathcal{S}) \leftarrow \mathcal{H}(\mathcal{S}) * h(m_i)$;
6: **end for**
7: $X_{\mathcal{S}} \leftarrow [g_1]^{\mathcal{H}(\mathcal{S})^{-1}}$;
8: **if** $\hat{e}(\sigma, g_2) = \hat{e}(X_{\mathcal{S}}, pk_u)$ **then**
9: $\quad B \leftarrow 1$
10: **else**
11: $\quad B \leftarrow 0$
12: **end if**
13: **return** B

3) Aggregation phase: The *aggregation* phase includes the agg and vrfagg algorithms, as detailed in Section III. The aggregating node executes the agg algorithm and derives the resulting signature that relies on set operations, namely union, subset, and intersection operations. The resulting signature, detailed in definition 3.2, must ensure the homomorphism and the correctness properties (cf. Equation 1 and Equation 2).

Theorem 1 shows our proposed signature scheme is homomorphic with respect to the union operation. The set-homomorphism properties, with respect to two sets S_i and S_j, is generated by H_i and H_j devices, respectively. Our scheme can easily be extended to support multiple sets of data records.

Theorem 1: **Union-homomorphic Signature**

Let stp, $\texttt{gen}_u$, sig, and vrf be the algorithms introduced by Algorithm 1, 2, 3, and 4. The agg algorithm resulting in an aggregate signature $\texttt{sig}(sk_i \circ sk_j, S_i \star S_j)$ presented in Equation 4 is defined as follows, such that $\star$ is the union operator, and $\circ$ is the minus operator:

$$\texttt{sig}(sk_i, S_i) \nabla \texttt{sig}(sk_j, S_j) = \frac{[\texttt{sig}(sk_i, S_i)]^{u'}}{[\texttt{sig}(sk_j, S_j)]^{v'}} \quad (6)$$

such that:

$$u' = u + v\frac{\mathcal{H}(S_i)}{\mathcal{H}(S_j)}, v' = v + u\frac{\mathcal{H}(S_j)}{\mathcal{H}(S_i)}$$

and u and v are the two unique integers that satisfy:

$$gcd(\mathcal{H}(S_i), \mathcal{H}(S_j)) = v\mathcal{H}(S_i) + u\mathcal{H}(S_j)$$

Proof 1: To prove Theorem 1, we must first express $\mathcal{H}(S_i \cup S_j)$, using $\mathcal{H}(S_i)$ and $\mathcal{H}(S_j)$, such that $\mathcal{H}(S_k) = \prod_{l \in [1, n_{S_k}]} h(s_l)$, where $k \in \{i, j\}$ and $n_{S_k} = |S_k|$ (i.e., the number of elements of a set S_k).

Lemma 1: Given two sets S_i and S_j and given a hash function $\mathcal{H}$ as defined in Algorithm 3, there exist two unique integers u and v such that:

$$\mathcal{H}(S_i \cup S_j)^{-1} = u\mathcal{H}(S_i)^{-1} + v\mathcal{H}(S_j)^{-1} \quad (7)$$

Proof 2: Given the hash function $\mathcal{H}$ as defined in Algorithm 3 the $\mathcal{H}(S_i \cup S_j)$ corresponds to:

$$\mathcal{H}(S_i \cup S_j) = lcm(\mathcal{H}(S_i), \mathcal{H}(S_j)) \quad (8)$$

In addition, knowing that lcm and gcd satisfy the following Equation:

$$lcm(\mathcal{H}(S_i), \mathcal{H}(S_j))gcd(\mathcal{H}(S_i), \mathcal{H}(S_j)) = \mathcal{H}(S_i)\mathcal{H}(S_j)$$

And using the Euclidean algorithm, there exist two unique integers u and v, such that:

$$gcd(\mathcal{H}(S_i), \mathcal{H}(S_j)) = v\mathcal{H}(S_i) + u\mathcal{H}(S_j)$$

Consequently, we have:

$$lcm(\mathcal{H}(S_i), \mathcal{H}(S_j))^{-1} = u\mathcal{H}(S_i)^{-1} + v\mathcal{H}(S_j)^{-1} \quad (9)$$

Based on Equation 8 and Equation 9, we prove Lemma 1.

In the following, we demonstrate that the `agg` algorithm satisfies the homomorphism and correctness properties. In this paper, we only prove for the union operation, but our scheme is easily extensible to the subset and set difference operations.

- *Proof of homomorphism –*

To prove that our `agg` ensures the homomorphism property, we have to express $\texttt{sig}(sk_i, S_i)$, referred to as σ_i, and $\texttt{sig}(sk_j, S_j)$, referred to as σ_j, with respect to both sk_i and sk_j.

$$\begin{aligned} Ⓢ &= {\sigma_i}^u {\sigma_j}^v \\ &= [{g_1}^{usk_i}]^{\mathcal{H}(S_i)^{-1}} [{g_1}^{vsk_j}]^{\mathcal{H}(S_j)^{-1}} \\ &= [{g_1}^{usk_i}]^{\mathcal{H}(S_i)^{-1}} [{g_1}^{vsk_j}]^{\mathcal{H}(S_j)^{-1}} \frac{[{g_1}^{vsk_i}]^{\mathcal{H}(S_j)^{-1}}}{[{g_1}^{vsk_i}]^{\mathcal{H}(S_j)^{-1}}} \\ &= {g_1}^{sk_i[u\mathcal{H}(S_i)^{-1} + v\mathcal{H}(S_j)^{-1}]} {g_1}^{v(sk_j - sk_i)\mathcal{H}(S_j)^{-1}} \end{aligned}$$

Similarly, we can express

$$Ⓡ = {\sigma_i}^u {\sigma_j}^v = {g_1}^{sk_j[u\mathcal{H}(S_i)^{-1} + v\mathcal{H}(S_j)^{-1}]} {g_1}^{u(sk_i - sk_j)\mathcal{H}(S_i)^{-1}}$$

Consequently, by dividing Ⓢ by Ⓡ, we obtain the following result:

$$\begin{aligned} {g_1}^{(sk_i - sk_j)[u\mathcal{H}(S_i)^{-1} + v\mathcal{H}(S_j)^{-1}]} &= \frac{{g_1}^{v(sk_j - sk_i)\mathcal{H}(S_j)^{-1}}}{{g_1}^{u(sk_i - sk_j)\mathcal{H}(S_i)^{-1}}} \\ {g_1}^{(sk_i - sk_j)[u\mathcal{H}(S_i \cup S_j)^{-1}]} &= \frac{{g_1}^{sk_i\mathcal{H}(S_i)^{-1}(u + v\frac{\mathcal{H}(S_i)}{\mathcal{H}(S_j)})}}{{g_1}^{sk_j\mathcal{H}(S_j)^{-1}(v + u\frac{\mathcal{H}(S_j)}{\mathcal{H}(S_i)})}} \\ \texttt{sig}(sk_i - sk_j, S_i \cup S_j) &= \frac{{\sigma_i}^{u'}}{{\sigma_j}^{v'}} \end{aligned}$$

where:

$$u' = u + v\frac{\mathcal{H}(S_i)}{\mathcal{H}(S_j)}, v' = v + u\frac{\mathcal{H}(S_j)}{\mathcal{H}(S_i)}$$

- *Correctness –*

This property shows how the verifier relies on the `vrfagg` in Algorithm 5 based on the aggregate signature $\frac{[\texttt{sig}(sk_i, S_i)]^{u'}}{[\texttt{sig}(sk_j, S_j)]^{v'}}$, the public keys pk_i and pk_j of nodes H_i and H_j, respectively, and the aggregate set $S_i \cup S_j$.

Based on Equation 2 and the properties of the bilinear function $\hat{e}$, we can write $\hat{e}({g_1}^{(sk_i - sk_j)[\mathcal{H}(S_i \cup S_j)^{-1}]}, g_2)$ denoted by $\mathcal{E}$, as follows:

$$\begin{aligned} \mathcal{E} &= \hat{e}({g_1}^{u'sk_i\mathcal{H}(S_i)^{-1} - v'sk_j\mathcal{H}(S_j)^{-1}}, g_2) \\ &= \hat{e}({g_1}^{u'sk_i\mathcal{H}(S_i)^{-1}}, g_2) \cdot \hat{e}({g_1}^{-v'sk_j\mathcal{H}(S_j)^{-1}}, g_2) \\ &= \hat{e}(\sigma_i, g_2)^{u'} \cdot \hat{e}(\sigma_j, g_2)^{-v'} \end{aligned}$$

As such, based on the `vrfagg` algorithm defined in Algorithm 5 and Equation 2, we deduce that $\texttt{vrfagg}(pk_i \Diamond pk_j, \texttt{sig}(sk_i - sk_j, S_i \cup S_j)) = 1$, such that $pk_i \Diamond pk_j = \frac{pk_i}{pk_j}$, if and only if:

$$\hat{e}(\sigma_i, g_2) = \hat{e}({g_1}^{\mathcal{H}(S_i)^{-1}}, pk_i), \text{ and}$$

$$\hat{e}(\sigma_j, g_2) = \hat{e}({g_1}^{\mathcal{H}(S_j)^{-1}}, pk_j)$$

Algorithm 5 `vrfagg` algorithm

1: **input:** the public keys pk_i and pk_j of nodes H_i and H_j, the aggregate set $S_i \cup S_j$ and the aggregate signature Σ
2: **output:** boolean value $B \in \{0, 1\}$
3: $\mathcal{H}(S_i \cup S_j) = 1$;
4: $n \leftarrow |S_i \cup S_j|$;
5: **for all** $k \in [1 \dots n]$ **do**
6: $\quad \mathcal{H}(S_i \cup S_j) \leftarrow \mathcal{H}(S_i \cup S_j) * h(s_k)$;
7: **end for**
8: $X \leftarrow [g_1]^{\mathcal{H}(S_i \cup S_j)^{-1}}$;
9: **if** $\hat{e}(\Sigma, g_2) = \hat{e}(X, \frac{pk_i}{pk_j})$ **then**
10: $\quad B \leftarrow 1$
11: **else**
12: $\quad B \leftarrow 0$
13: **end if**
14: **return** B

This proves the correctness of our proposed aggregate signature, with respect to Definition 3.2.

V. Security Discussion

A. Unforgeability

As discussed in Section II-B, the unforgeability requirement prevents external adversaries from generating a new valid signature based on new generated data.

Obviously, $\mathcal{A}$ tries a forgery attack against the CDH and co-CDH assumptions considering the simple signature σ and the aggregate signature Σ are both products of an accumulator over the set elements and a group element associated to the secret key of the signer. Knowing these two group elements, namely the accumulator and the group element associated with the secret key of the signer, are both required for deriving the corresponding signature, $\mathcal{A}$ is led to break the co-CDH and CDH assumptions. The Exp^{unf} is then considered with respect

to the CDH-assumption, as it is considered stronger than the co-CDH assumption [13].

The complexity of the CDH assumption has been studied in [13] and was demonstrated to be hard to solve; i.e., a (t, ϵ) CDH group is a group for which the $\mathbf{Adv}(\mathcal{A}, t) \leq \epsilon$ for every PPT adversary running in a time t. Therefore, our signature scheme is unforgeable.

B. Privacy

The privacy-preserving property covers the privacy of contents, where the adversary should not be able to guess the origin and the content of the removed parts from signed data flow, and the privacy of users, where an intruder should not be able to link each signed part of an aggregate data stream to its respective generator.

Generally, the attacker $\mathcal{A}$ tries to distinguish between two honestly derived signatures by linking each simple signature of an aggregate signature Σ to its related generator, having only access to Σ.

The aggregate signature Σ results in merging two signatures σ_b and $\sigma_{\bar{b}}$, where $b = (j, k)$, such that j refers the set index (i.e., $j \in \{0, 1\}$) and k to the index of the private key (i.e., $k \in \{1, 2\}$). Each single signature $\sigma_{(j,k)}$, generated by $\mathcal{C}$ using the private key sk_k, is based on a randomly chosen set S_j. As such, both signatures σ_b and $\sigma_{\bar{b}}$ are identically distributed in both cases. Thus, the attacker $\mathcal{A}$ is relying on a *left-or-right* oracle with respect to the CDH assumption. Hence, he cannot distinguish the oracle's outputs better than flipping a coin.

Similarly, the aggregate signature Σ generated based on two signatures associated with two different private keys, sk_k (i.e., $k \in \{1, 2\}$), is uniformly distributed. Hence, two aggregate signatures derived based on two different combinations b and $\bar{b}$ are statistically indistinguishable, and the probability of predicting b is $\frac{1}{2}$ with respect to the CDH assumption.

As such, the $\mathcal{A}$ cannot distinguish two different aggregate signatures with a probability greater than $\frac{1}{4}$. Therefore, our scheme provides the privacy property.

VI. Conclusion

The need for secure mechanisms that enable the end receiver to efficiently verify the authenticity of aggregated data streams and the emergence of homomorphic signatures led us to present a cryptographic mechanism that enables efficient aggregation of signed data. In this paper, we proposed a new lightweight aggregate signature scheme based on set-homomorphic relations in distributed environments. Our construction permits distribution of the verification process among multiple IoT nodes. We proved its homomorphism and correctness. In addition, our construction preserves the privacy of non-end nodes such that an external adversary cannot distinguish between honestly derived signatures by linking each simple signature of an aggregate signature to its respective generator.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant ACI-1547467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[2] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "A survey of two signature aggregation techniques," *CryptoBytes*, vol. 6, p. 2003, 2003.

[3] ——, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003.

[4] L. Wang, L. Wang, Y. Pan, Z. Zhang, and Y. Yang, "Discrete logarithm based additively homomorphic encryption and secure data aggregation," *Inf. Sci.*, vol. 181, no. 16, pp. 3308 – 3322, Aug. 2011.

[5] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Cryptographers Track at the RSA Conference*. Springer, 2002, pp. 244–262.

[6] G. Ateniese, D. H. Chou, B. De Medeiros, and G. Tsudik, "Sanitizable signatures," in *European Symposium on Research in Computer Security*. Springer, 2005, pp. 159–177.

[7] D. Slamanig and S. Rass, "Generalizations and extensions of redactable signatures with applications to electronic healthcare," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2010, pp. 201–213.

[8] D. Boneh and D. M. Freeman, "Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures," in *International Workshop on Public Key Cryptography*. Springer, 2011, pp. 1–16.

[9] S. Canard, A. Jambert, and R. Lescuyer, "Sanitizable signatures with several signers and sanitizers," in *International Conference on Cryptology in Africa*. Springer, 2012, pp. 35–52.

[10] K. Samelin, H. C. Pöhls, A. Bilzhause, J. Posegga, and H. De Meer, "Redactable signatures for independent removal of structure and content," in *International Conference on Information Security Practice and Experience*. Springer, 2012, pp. 17–33.

[11] G. Neven, "A simple transitive signature scheme for directed trees," *Theoretical Computer Science*, vol. 396, no. 1-3, pp. 277–282, 2008.

[12] A. Bagherzandi and S. Jarecki, "Identity-based aggregate and multi-signature schemes based on rsa," in *International Workshop on Public Key Cryptography*. Springer, 2010, pp. 480–498.

[13] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ser. ASIACRYPT '01, 2001.