# Secure Data Aggregation Protocol for Sensor Networks

Presented By : Kavit Shah

Major Advisor : Brian King

Thesis Committee Members : Paul Salama, Mohamed El-Sharkawy, Sangkook Lee

## Overview

## Introduction

**Ubiquitous Computing -** is a scenario in which computing is everywhere.

**Internet of Things -** is a system where the Internet is connected to physical world via ubiquitous sensors.

**Ad-hoc Networking -** is a local network of sensors formed by peer-to-peer communications.

**Sensor Networks -** Collectively, we refer these concept as Sensor Networks.

## Sensor Networks

In sensor networks, thousands of sensors may interact with each other and collects raw data.

The data is processed by computationally powerful machine (the base station).

Then the base station converts data into information.

Based on the derived information an important action is taken.

## Sensor Networks Applications

**Military -** enemy tracking, battle filed surveillance or target classification.

**Environmental -** to monitor geographical location without much human intervention.

**Health Care -** to monitor patients around the clock, send reminders to doctors and nurses.

**Sustainable Mobility -** to build digitally connected and coordinated vehicles.

## Consequences of Sensor Failures

Speed sensor failure led to crash of Air France flight - Airbus A330 on 1st June 2009.

**For nearly a minute**, as the speed sensors jumped, the pilot was not present in the cockpit.

The pilots could not reclaim control as the plane dropped out of the sky at a rate of **10,000 feet per minute**.

The flight plunged into the Atlantic nose-up after its three-and-a-half minute freefall , killing all **228** on board.

## Resource Constrains in Sensor Network

**Physical Limitations -** often deployed in open, hostile and unattended environments.

**Hardware Limitations -** due to lower manufacturing cost of sensor nodes, they have low speed processor, limited storage, a short range trans receivers.

**Transmission Medium -** wireless networks have approximately $10^6$ **times higher bit error rate (BER)** than wired networks which causes frequent link loss and then path loss.

**Mobility -** network topology is dynamic, topology changes due to link failure, node failure or bandwidth optimization.

# Cryptographic Tools

## Hash Functions

A hash function takes a message as its input and outputs a fixed length message called hash code, creating a digital fingerprint of the message.

**Compression** A function $h$ maps an input $x$ of arbitrary finite bitlength, to an output $h(x)$ of fixed bitlength $n$.

**Preimage resistance** For all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage $x'$ such that $h(x') = y$ where $y$ is given whose corresponding input is not known.

**2nd-preimage resistance** It is computationally infeasible to find any second input which has the same output as any specified input, i.e, given $x$, to find a 2nd-preimage $x' \neq x$ such that $h(x') = h(x)$.

**Collision resistance** It is computationally to find any two distinct inputs $x, x'$ which hash to the same output, i.e., such that $h(x) = h(x')$.

SHA-256, is a 256-bit hash and provides $128$ bits of security against collision attacks.

## Message Authentication Codes

A Message Authentication Code (MAC) is a family of hash functions parameterized by a secret key $k$.

**Ease of computation** For a known function $h_k$, given a value $k$ and an input $x$, $h_k(x)$ is easy to compute.

**Compression** A function $h_k$ maps an input $x$ of arbitrary finite bitlength to an output $h_k(x)$ of fixed length $n$.

**Computation-resistance** Given a description of the function family $h$, for every fixed allowable value of $k$ (unknown to an adversary), given zero or more text-MAC pairs ($x_i, h_k(x_i)$), it is computationally infeasible to compute any text-MAC pair ($x, h_k(x)$) for any new input $x \neq x^{'}$ (including possibly for $h_k(x) = h_k(x_i)$ for some $i$). If computation-resistance does not hold, a MAC algorithm is subject to MAC-forgery.

## Digital Signatures

A digital signature is a cryptographic scheme for demonstrating the authenticity of a digital message.

A valid digital signature gives a recipient strong reason to believe that the message was created by a known sender, such that the sender can not deny having sent the message (**authentication and non-repudiation**) and that the message was not altered in transit (**integrity**).

It is a 5-tuple scheme ($\mathcal{M}, \mathcal{K}, \mathcal{G}, \mathcal{S}, \mathcal{V}$)

For any secret key $s_k \in \mathcal{K}$ and any $m \in \mathcal{M}$, the message $m$ is signed using key $s_k$ as follows:

$$s = \text{Sign}_{s_k}(m) \tag{1}$$

For any $s_k$ let $p_k$ denote public key and for all $m \in \mathcal{M}$ and $s \in \mathcal{S}$, $s$ as follows:

$$\text{Verify}_{p_k}(m, s) = \begin{cases} \textbf{true} \text{ with probability of 1} & \text{if } s = \text{Sign}_{s_k}(m) \\ \textbf{false} \text{ with overwhelming probability} & \text{if } s \neq \text{Sign}_{s_k}(m) \end{cases} \tag{2}$$

where the probability space is determine by the $\mathcal{M}, \mathcal{S}, \mathcal{K}, \mathcal{K}^{'}$ and perhaps the signing and verification algorithms. The “overwhelming probability” for the signature scheme determines the probability that the scheme allows for a forgery.

## Summary of Cryptography Tools

Three different integrity-protection mechanisms HASH, MAC, Signature are summarized here.

| | Who can generate it | Who can verify it |
|---|---|---|
| Hash | Everyone | Everyone |
| MAC | Holders of secret | Holders of secret |
| Signature | Holder of secret | Everyone |

## Data Aggregation

The idea of Data Aggregation is to compress the data coming from different sources enroute eliminating redundancy, minimizing the number of transmissions, thus saving energy and increasing the longevity of the network.

This paradigm shifts the focus from the traditional **address centric** approaches for networking (finding short routes between pairs of addressable end nodes) to a more **data-centric** approach (finding routes from multiple sources to a single destination that allows in-network consolidation of redundant data).

## Definition

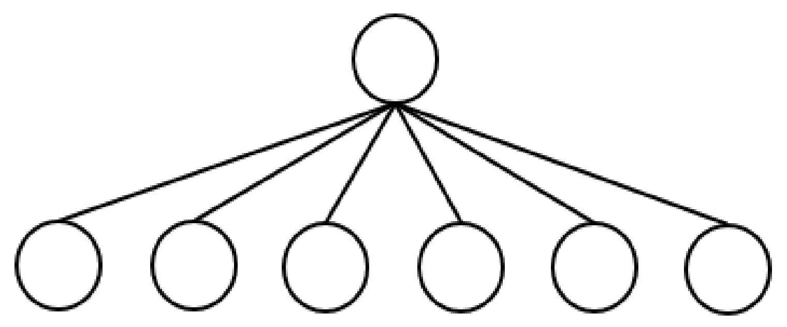
Consider a sensor network with $n$ sensors collecting data and the base station, processes data into information.

This can be represented as $f(x_1, x_2, ..., x_n)$ where $x_1, x_2, ..., x_n$ represent the sensor readings.

The goal is to compute the information as follows :

$$y = f(x_1, x_2, ..., x_n)$$

## Data Aggregation using SUM



The root in the network, receives the following sensor data $S_1(10), S_2(14), S_3(12), S_4(15), S_5(11), S_6(17)$ and has its own data $S_0(15)$.

The root node aggregates these seven data and creates an aggregated result as follows:

$$S = \sum_{i=0}^{6} S_i \tag{3}$$

Now, root has to send only one data to its parent instead of seven.

## Lossy Data Compression

Lossy data compression schemes produces a compressed file from which only an **approximation** to the original information can be recovered. Much higher compression ratios are possible.

Our aggregation protocol is **lossy data compression** as the base station receives an aggregated sensor data and can not recover the original sensor data.

## Secure Hierarchical In-network Data Aggregation

SHIA is a light weight protocol for providing data-integrity to in-network data aggregation scheme.

It is designed for general networks with single or multiple adversaries.

Our work enhances SHIA, by making it communication efficient, adds new security services to the protocol, achieves similar security goals with non-resilient aggregation functions and efficient ways of analyzing the protocol.

## Network Assumptions

A multi hop network with a set $S = \{S_1, ..., S_n\}$ of $n$ nodes where all sensor nodes are alive and reachable.

The network is organized in a **rooted tree topology**.

The **trusted base station** resides outside of the network and the network and the network topology.

The base station can directly communicate with every node in the network using **authenticated broadcast**.

All the wireless communications is **peer-to-peer** and we do not consider the **local wireless broadcast**.

Each sensor node $I$ has a unique **ID** and shares a unique secret symmetric key $\mathsf{sk}_\mathsf{I}$ with the base station.

The sensor nodes are capable of doing **symmetric and asymmetric** key encryption and decryption.

## Attacker Model

We consider a model with **polynomially bounded** adversary (polynomial in terms of the security parameter), which has a complete control over some of the sensor nodes in the network.

We focus on **stealthy attacks**, where the goal of an adversary is to make the base station accept false aggregation results, which are significantly different from the true results determined by the measured values, while not being detected by the base station.

We do not consider **denial-of-service (DoS)** and various **passive attacks**.

## SUM Aggregate Algorithm

Here, the aggregate function $f$ is summation meaning that we want to compute $a_1 + a_2 + ... + a_n$, where $a_i$ is the sensor reading of the node $i$.

The algorithm has the following three main phases.

**Query Dissemination,** initiates the aggregation process.

**Aggregate Commit,** initiates the commitment tree generation process.

**Result Checking,** initiates the distributed, interactive verification process.

## Aggregation Tree

We require the prior constructed rooted aggregation tree.

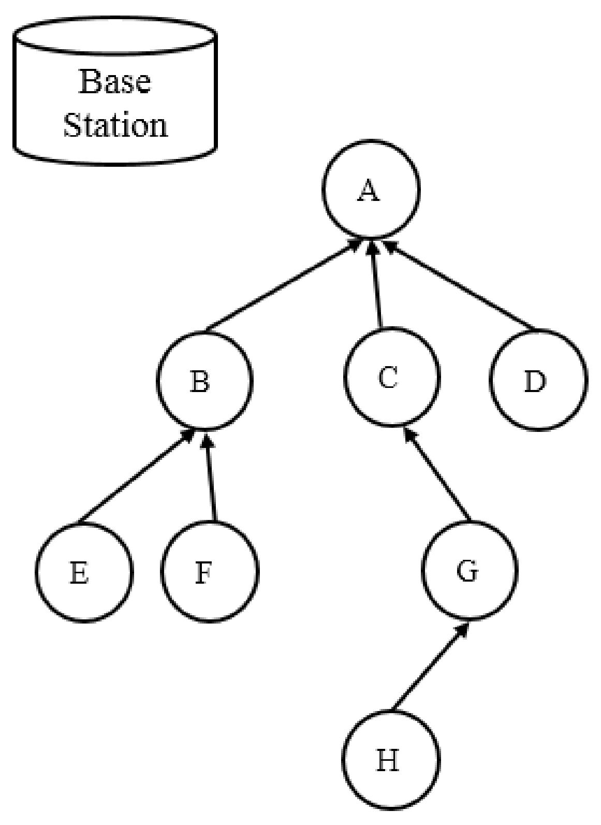


Figure 1: Aggregation Tree

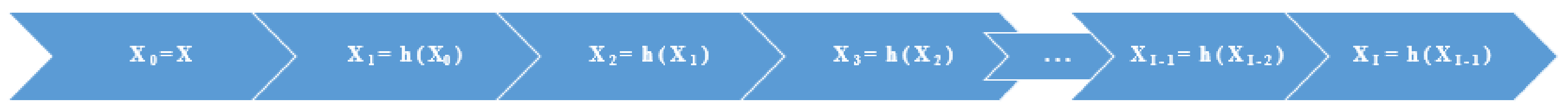
## Query Dissemination

The base station broadcasts the query request message with the query nonce $N$ in the aggregation tree.

SHIA uses **hash chain** to generate new nonce for each query.

A hash chain is constructed by repeatedly evaluating a pre-image resistant hash function $h$ on some initial random value, the final value (or “anchor value”) is preloaded on the nodes in the network.

The base station uses the pre-image of the last used value as the nonce for the next broadcast.



Hash Chain.

For example, if the last known value of the hash chain is $h^i(X)$, then the next broadcast uses $h^{i-1}(X)$ as the nonce; $X$ is the initial random value.

When a node receives a new nonce $N^{'}$, it verifies that $N^{'}$ is a precursor to the most recently received (and authenticated) nonce $N$ on the hash chain, i.e., $h^i(N^{'}) = N$ for some $i$ bounded by a fixed $k$ of number of hash applications.

## Aggregate Commit

**Defn:** A commitment tree is a tree where each vertex has an associated label representing the data that is passed on to its parent: $<$count, value, complement, commitment$>$

The sensor node $A$ creates the label for the leaf vertex $A_0$ is given as follows :

$$A_0 = < A_{count}, A_{value}, \overline{A_{value}}, A_{commitment} >$$

Internal node $I$ has child vertices: $A_1, A_2, \ldots, A_q; A_i = < A_{icount}, A_{ivalue}, \overline{A_{ivalue}}, A_{icommitment} >$.
Then,

$$I_j = < I_{jcount}, I_{jvalue}, \overline{I_{jvalue}}, I_{jcommitment} >$$
$$I_{jcount} = \sum A_{icount}$$
$$I_{jvalue} = \sum A_{ivalue}$$
$$\overline{I_{jvalue}} = \sum \overline{A_{ivalue}}$$
$$I_{jcommitment} = H[N || I_{jcount} || I_{jvalue} || \overline{I_{jvalue}} || A_1 || A_2 || \ldots || A_q]$$

## Commitment Tree Generation Algorithm

Leaf nodes in the aggregation tree originate a single-vertex commitment forest.

Internal node $I$ originates a similar single-vertex commitment forest and receives commitment forests from its children.

$I$ wishes to combine $q$ commitment forests $F_1, \ldots, F_q$.

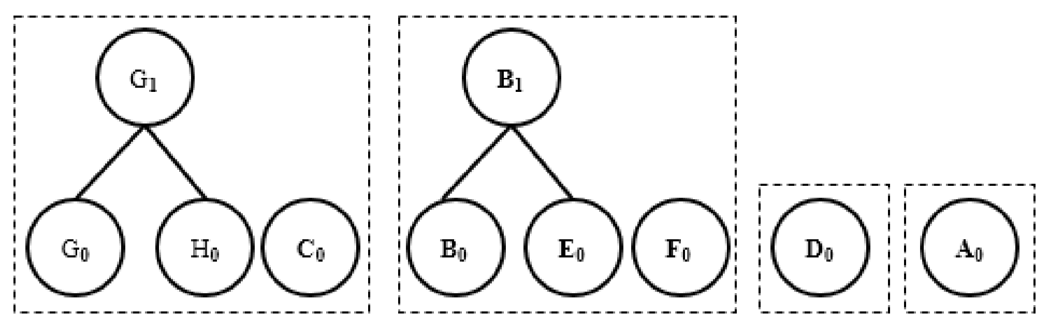
The intermediate result be $F = F_1 \cup \ldots \cup F_q$, and repeat the following until no two trees are the same height in $F$.

Let $h$ be the smallest height such that more than one tree in $F$ has height $h$.

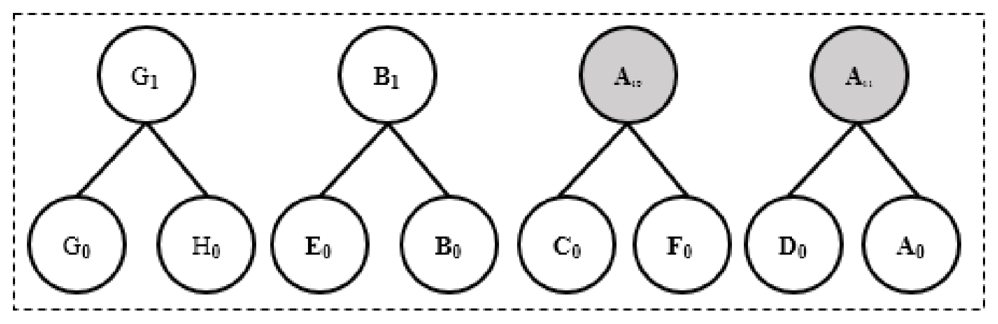
Find two commitment trees $T_1$ and $T_2$ of height $h$ in $F$, and merge them into a tree of height $h + 1$ by creating a new vertex that is the parent of both the roots of $T_1$ and $T_2$.

## Commitment Tree Generation Example

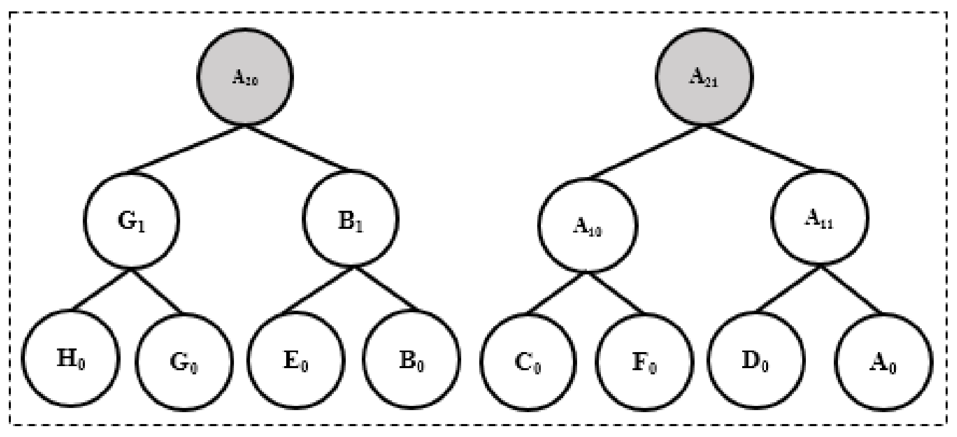
The commitment tree generation for the root $A$ of Figure 1 is given as follows:



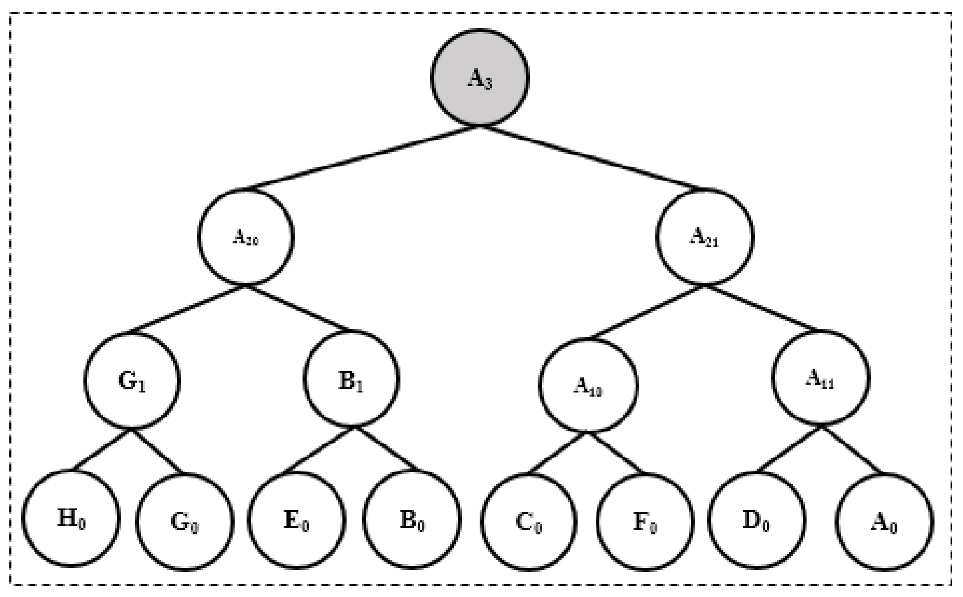
$A$'s forest impacted by the forests received from $B$, $C$, $D$.



$A$'s forest after first merge.



$A$'s forest after second merge.



$A$'s outgoing forest.

$A_3 =< 8, A_{3value}, \overline{A_{3value}}, H(N||8||A_{3value}||\overline{A_{3value}}||A_{20}||A_{21}) >$

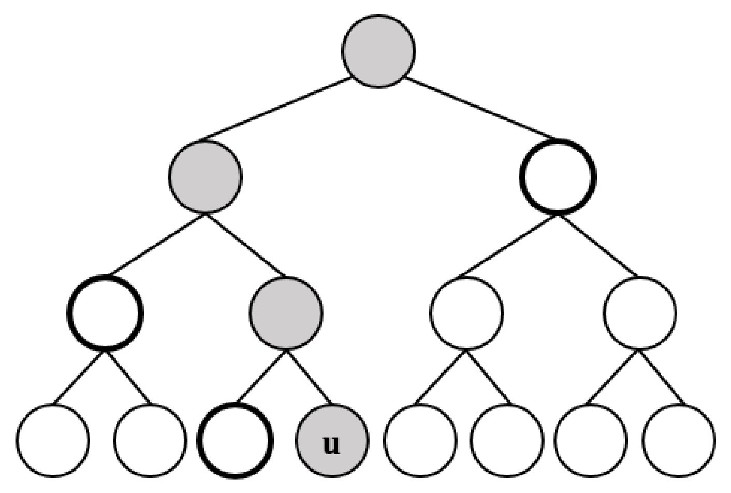
$A_{3value} = A_{20value} + A_{21value}$

$\overline{A_{3value}} = \overline{A_{20value}} + \overline{A_{21value}}$

## Result Checking

**Distributing Final Commitment Values** The base station broadcasts all the received commitment labels to the entire network using authenticated broadcast.

**Distributing Off-path Values** The set of **off-path vertices** for a vertex $u$ in a tree is the set of all the siblings of each of the vertices on the path from $u$ to the root of the tree that $u$ is in (the path is inclusive of $u$).



Off-path vertices of $u$ are highlighted in bold.

**Verification of Contribution** The leaf vertex calculates the root label using received off-path labels.

It compares the the calculated label with the received label from the base station.

If those two labels match then it proceeds to the next step with Acknowledgment (ACK) message or with Negative Acknowledgment (NACK) message.

**Collection of Authentication Codes** The authentication code for sensor node $A$ with ACK and NACK message is given as follows :

$$\mathsf{MAC}_{\mathsf{sk}_\mathsf{A}}(\mathsf{N}||\mathsf{ACK})$$
$$\mathsf{MAC}_{\mathsf{sk}_\mathsf{A}}(\mathsf{N}||\mathsf{NACK})$$

After receiving the authentication codes from all of its children it does XOR operation on all the authentication codes including its own authentication code and before sending to its parent.

**Verification of confirmations** The base station computes the following:

$$\bigoplus_{i=1}^{n} \mathsf{MAC}_{\mathsf{sk}_\mathsf{i}}(\mathsf{N}||\mathsf{ACK})$$

Then it compares the computed code with the received code. If those two codes match, then the base station accepts the aggregation result.

## Our Protocol

### Data Item

A commitment tree is a binary tree where each vertex has an associated data-item representing the data that is passed on to its parent.

$$< id, count, value, commitment >$$

Each sensor node creates its own data-item. For example, sensor node $A$ creates its data-item $A_0$.

$$A_0 = < A_{id}, 1, A_{value}, H(N||1||A_{value}) >$$

where $A_{id}, A_{value}$ is the unique ID and sensor reading of the node $A$. The count is $1$ as there is only vertex in the subtree rooted at $A$, $H$ is the collision resistant hash function, and $N$ is the query nonce.

## Signing and Verification of the Data-item

Each sensor node sends the signature of its data-item signed by itself using its own secret key.

$$\mathsf{S} = \mathsf{Sign}_{\mathsf{sk_A}}(A_0)$$

Table: Digital Certificate

| |
|---|
| Unique ID of the sensor node |
| Public key of the sensor node |
| Certification Authority's name |
| Certification Authority's digital signature |

$$\mathsf{Verify}_{\mathsf{pk_A}}(A_0, \mathsf{S}) = \begin{cases} \textbf{true} \text{ with probability of 1} & \text{if } \mathsf{S} = \mathsf{Sign}_{\mathsf{sk_A}}(A_0) \\ \textbf{false} \text{ with overwhelming probability} & \text{if } \mathsf{S} \neq \mathsf{Sign}_{\mathsf{sk_A}}(A_0) \end{cases}$$

## Commitment Payload

A **commitment payload** is a set of data-items of the root vertices of the trees with their respective signatures in the outgoing commitment forest and an additional signature for the transmission.

The **transmit payload** is the concatenation of all the data-items in the commitment payload.

## Commitment Payload Example

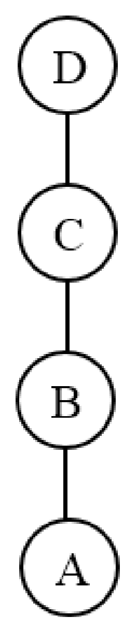


Figure 3: Palm Aggregation Tree

$$A_{pay} = < A_0, \mathsf{Sign}_{\mathsf{sk_A}}(A_0), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) > \; where \; A_\tau = < A_0 >$$

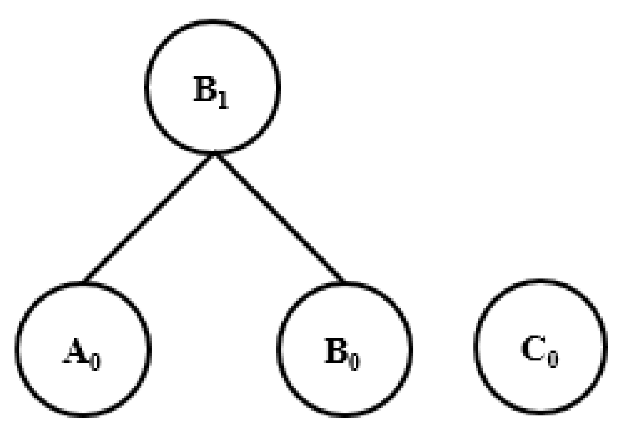


Figure 4: Commitment Payload of C

$C_{pay} = < C_0, \mathsf{Sign}_{\mathsf{sk_C}}(C_0), B_1, \mathsf{Sign}_{\mathsf{sk_B}}(B_1), \mathsf{Sign}_{\mathsf{sk_C}}(C_\tau) > \ where\, C_\tau =< C_0||B_1 >$
$C_0 = < C_{id}, 1, C_{value}, H(N||1||C_{value}) >$
$B_1 = < B_{id}, 2, B_{1value}, H(N||2||B_{1value}||A_0||B_0) >;\ B_{1value} =\ B_{value} + A_{value}$

## FSwRD vs FSwoRD

Forwarding Signatures With Resigning Data-Item (FSwRD)

$$C_{pay} = < C_0, \mathsf{Sign}_{\mathsf{sk_C}}(C_0), B_1, \mathsf{Sign}_{\mathsf{sk_C}}(B_1), \mathsf{Sign}_{\mathsf{sk_C}}(C_\tau) > \ where \, C_\tau = < C_0 || B_1 >$$

Forwarding Signatures Without Resigning Data-Item (FSwoRD)

$$C_{pay} = < C_0, \mathsf{Sign}_{\mathsf{sk_C}}(C_0), B_1, \mathsf{Sign}_{\mathsf{sk_B}}(B_1), \mathsf{Sign}_{\mathsf{sk_C}}(C_\tau) > \ where \, C_\tau = < C_0 || B_1 >$$

## Analogy for FSwRD vs FSwoRD

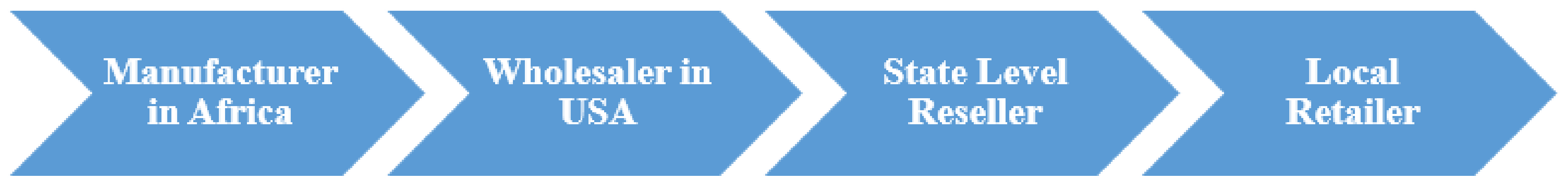


Figure 1: Diamond Supply Chain.

## Security Benefits of Signatures

The signature allows the parent node to verify the **authenticity** of the sensor node and assures the **integrity** of the data-item.

It allows the sender to have the proof for the sent data-item and the receiver to have the proof for the received data-item, providing the security service of **non-repudiation**.

The digital signature depends on the message so the parent node can not reuse the signature for other messages in the future, protecting the network against the **replay attacks**.

The signature of the transmit-payload is like the signature for the transmission, assuring none of the data-items in its payload have been left stranded.

## Aggregate Commit

This phase creates the commitment tree for the given aggregation tree.

## Commitment Tree Generation

Leaf nodes in the aggregation tree construct and send their payload to their parents in the aggregation tree.

Each internal node in the aggregation tree constructs their leaf vertex.

Internal node verifies all the received signatures then it merges all the data-items with same count value from its forest.

It merges two data-items by creating a new data-item with count value incremented by one and whose value is the addition of value field of the previous two data-items.

For example, in Figure 0, the root $A$ receives payloads from each of its children.

We describe the payload generation process for nodes $D, B, C$ and $A$ in order.

## Example Continue: D’s Payload Generation

The node $D$ constructs its payload and sends $D_{pay}$ to its parent $A$.

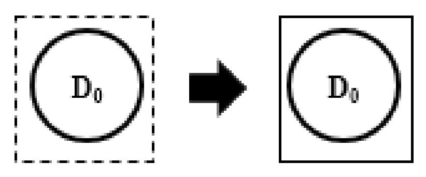


Figure 2: Transformation from $D$’s forest to its payload. Each dashed-line box shows forest and solid-line box shows payload of the respective sensor node.

$$D_{pay} = < D_0, \mathsf{Sign}_{\mathsf{sk_D}}(D_0), \mathsf{Sign}_{\mathsf{sk_D}}(D_\tau) >;\ where\ D_\tau = < D_0 >$$

$$D_0 = < D_{id}, 1, D_{value}, H(N||1||D_{value}) >$$

## Example Continue: B’s Payload Generation

The node $B$ constructs its payload from its forest which consists of payloads received from $E$ and $F$.

Then node $B$ sends $B_{pay}$ to its parent $A$.

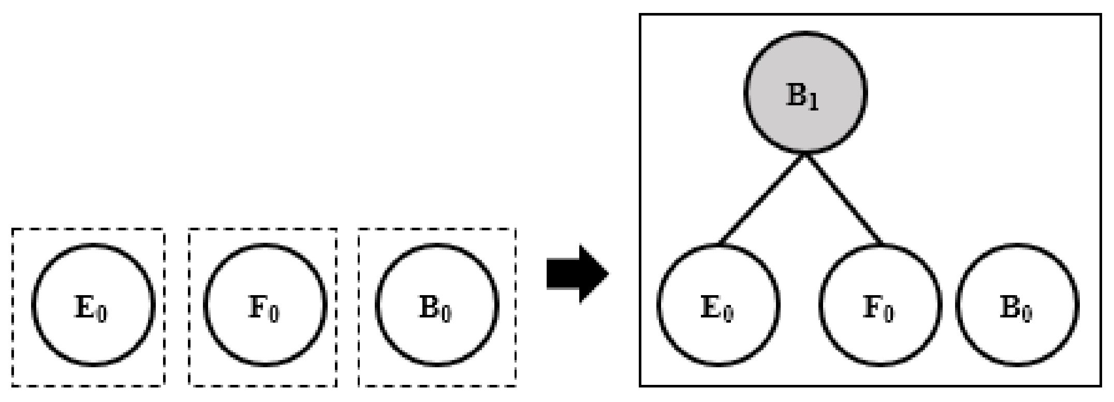


Figure 3: Transformation from $B$’s forest to its payload.

$$B_1 = < B_{id}, 2, B_{1value}, H(N||2||B_{1value}||E_0||F_0) >;\ B_{1value} = E_{value} + F_{value}$$

$$B_{pay} = < B_0, \mathsf{Sign}_{\mathsf{sk_B}}(B_0), B_1, \mathsf{Sign}_{\mathsf{sk_B}}(B_1), \mathsf{Sign}_{\mathsf{sk_B}}(B_\tau) > \ where\ B_\tau = < B_0||B_1 >$$

## Example Continue: C’s Payload Generation

The node $C$ constructs its payload from its forest which consists of payloads received from $G$.

Then node $C$ sends $C_{pay}$ to its parent $A$.

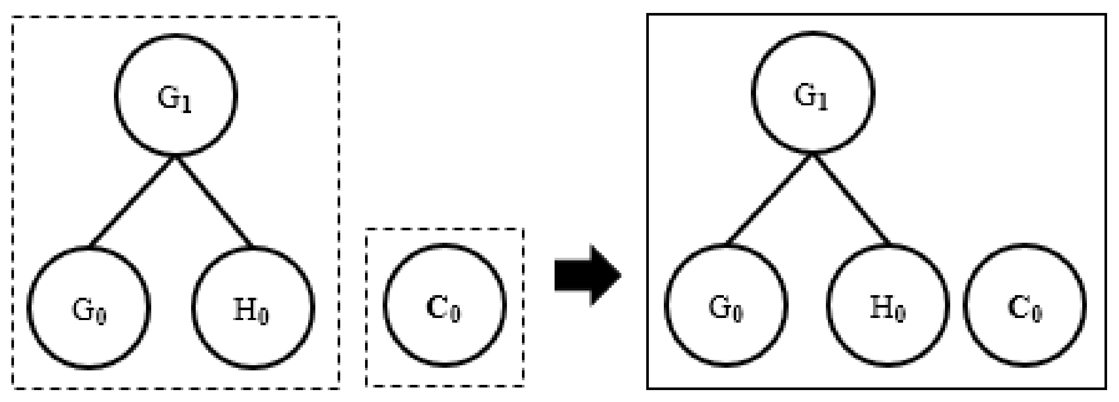


Figure 4: $C$’s forest aggregation creating its payload.

$$G_{pay} = < G_1, \textsf{Sign}_{\textsf{sk}_\textsf{G}}(G_1), \textsf{Sign}_{\textsf{sk}_\textsf{G}}(G_\tau) > \ where\, G_\tau = < G_0 || H_0 >$$

$$C_{pay} = < C_0, \textsf{Sign}_{\textsf{sk}_\textsf{C}}(C_0), G_1, \textsf{Sign}_{\textsf{sk}_\textsf{C}}(G_1), \textsf{Sign}_{\textsf{sk}_\textsf{C}}(C_\tau) > \ where\, C_\tau = < C_0 || G_1 >$$

## Example Continue: A's Payload Generation

The root node of the aggregation tree $A$ receives the payloads from $B, C$ and $D$ respectively.

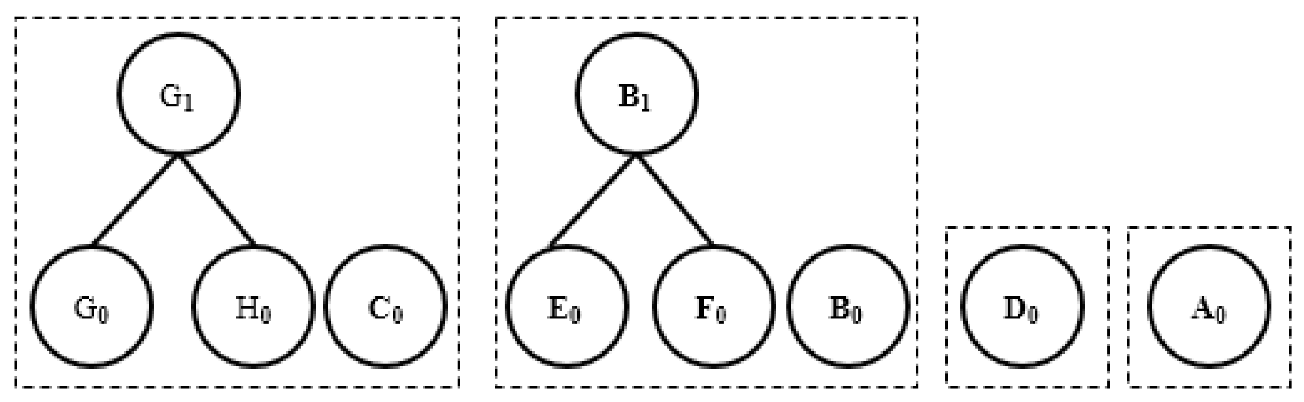


Figure 5: $A$'s forest: $A$ receives three payloads from $C, B$ and $D$

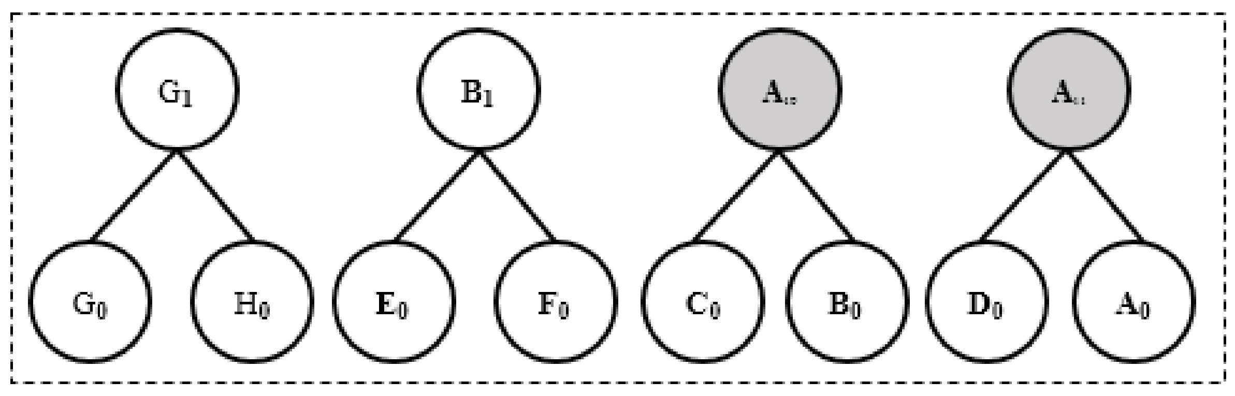


Figure 6: $A$'s forest: after first merge

$$A_{10} = < A_{id}, 2, A_{10value}, H(N||2||A_{10value}||B_0||C_0) >;\ A_{10value} =\ B_{value} + C_{value}$$
$$A_{11} = < A_{id}, 2, A_{11value}, H(N||2||A_{11value}||D_0||A_0) >;\ A_{11value} =\ D_{value} + A_{value}$$

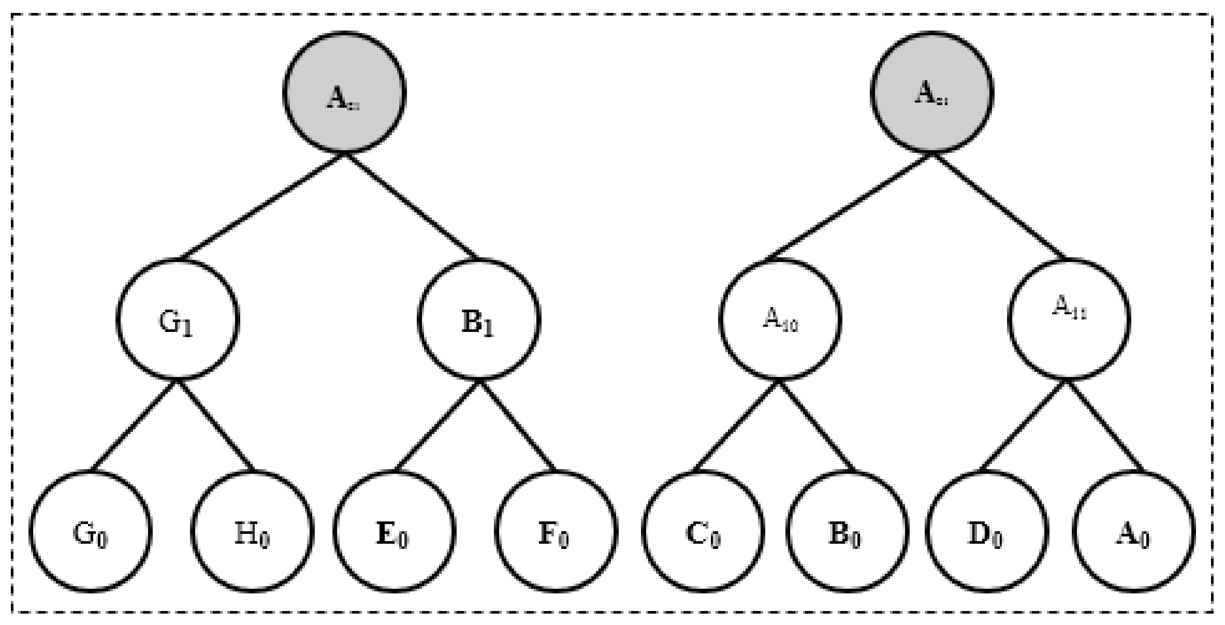


Figure 7: $A$'s forest: after second merge

$A_{20} = < A_{id}, 4, A_{20value}, H(N||4||A_{20value}||G_1||B_1) >;\ A_{20value} =\ G_{1value} + B_{1value}$

$A_{21} = < A_{id}, 4, A_{21value}, H(N||4||A_{21value}||A_{10}||A_{11}) >;\ A_{21value} =\ A_{10value} + A_{11value}$

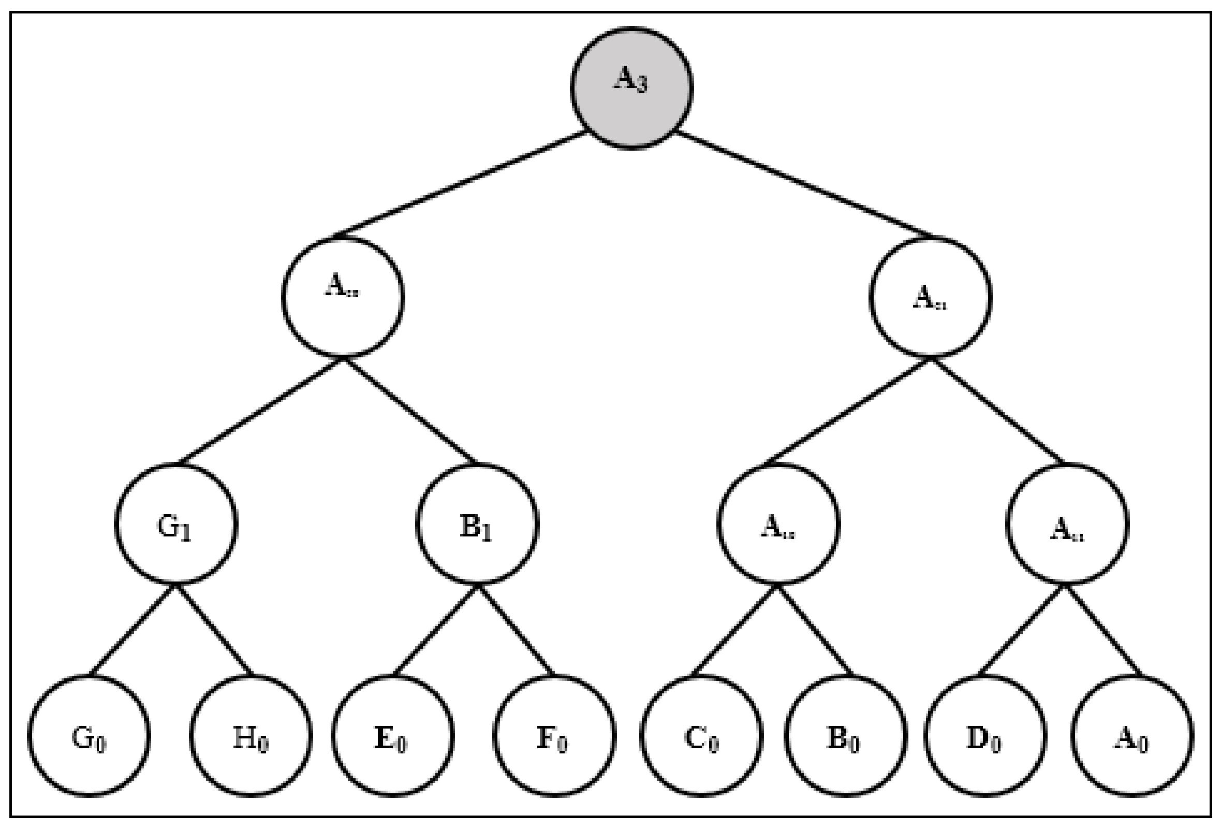


Figure 8: $A$'s payload : $A$ sends $A_3$ to the base station.

$A_3 =< A_{id}, 8, A_{3value}, H(N||8||A_{3value}||A_{20}||A_{21}) >;\ A_{3value} =\ A_{20value} + A_{21value}$
$A_{pay} =< A_3, \mathsf{Sign}_{\mathsf{sk_A}}(A_3), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >\ where\ A_\tau =< A_3 >$

## Result Checking

This phase requires that all the sensor nodes verify their individual contributions to the final aggregate value.

If there is any inconsistency in the aggregation process then with the help of the base station, trace down the node responsible for inconsistency.

It has the following major steps :

**Dissemination of Final Payload by the Base Station**

**Dissemination of Off-Path Values**

**Verification of Inclusion**

**Collection of Authentication Codes**

**Verification of Authentication Codes**

**Detecting An Adversary**

## Dissemination of Final Payload by the Base Station

The base station broadcasts all the data-items in the payload to entire network using **authenticated broadcast**.

The base station receives only one data-item $A_3$ in the payload sent by $A$.

The base station broadcasts $B_{pay}$ to entire network.

$$\mathsf{B}_{pay} = < A_3, \mathsf{Sign}_{\mathsf{sk_B}}(A_3), \mathsf{Sign}_{\mathsf{sk_B}}(\mathsf{B}_\tau) > \; where \, \mathsf{B}_\tau = < A_3 > .$$

## Dissemination of Off-Path Values

In Figure 8, $A_{10}$ has two children $C_0$ and $B_0$.

$A_{10}$ also receives $A_{11}$ and $A_{20}$ from its parent $A_{21}$.

$A_{10}$ ( which is sensor node $A$ in aggregation tree ) sends the following off-path values to $C$ and $B$ respectively.

$$< B_0, \mathsf{Sign}_{\mathsf{sk_A}}(B_0), A_{11}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{11}), A_{20}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{20}), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >$$

$$< C_0, \mathsf{Sign}_{\mathsf{sk_A}}(C_0), A_{11}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{11}), A_{20}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{20}), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >$$

## FSwoRD

$$< B_0, \mathsf{Sign}_{\mathsf{sk_B}}(B_0), A_{11}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{11}), A_{20}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{20}), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >$$

$$< C_0, \mathsf{Sign}_{\mathsf{sk_C}}(C_0), A_{11}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{11}), A_{20}, \mathsf{Sign}_{\mathsf{sk_A}}(A_{20}), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >$$

In FSwRD, all the leaf vertices need to know only one certificate as they receive data-items signed by their parent vertex.

In FSwoRD, all the leaf vertices might need to know $\log l$ certificates, where $l$ is the number of leaf-vertices in commitment tree.

## Significance of Commitment Filed

The commitment filed provides data-integrity and helps us detecting any *malicious activity* in the network. The signatures infrastructure eventually we can detect an adversary.

If an internal vertex simply **forwards incorrect data-item** then the relevant leaf vertex will complain, as they will not be able to derive the data-item received using authenticated broadcast from the base station.

If an internal vertex **changed the data-item** while creating commitment tree and sending the incorrect off-path values to compensate discrepancy.

## Malicious Activity

Suppose the vertices in the commitment tree have the data-items defined as follows :

$$A_0 = < A_{id}, 1, 10, H(N||1||10) >$$
$$B_0 = < B_{id}, 1, 20, H(N||1||20) >$$
$$C_1 = < C_{id}, 2, 30, H(N||2||30||A_0||B_0) >$$

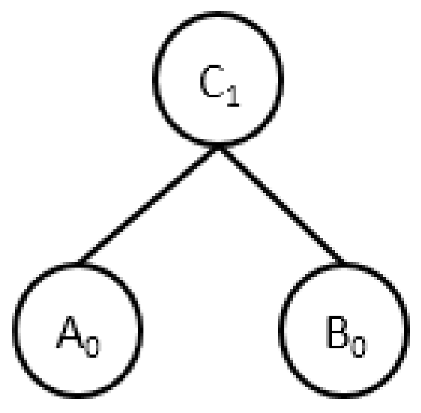


Figure 9: Smallest Possible Commitment Tree

Suppose $C$ changes $A_0$ and $B_0$ to $A'_0$ and $B'_0$.

$C$ can send either $C'_1$ or $C''_1$ to the base station.

To compensate for the discrepancy, $C$ constructs $B''_0$ and $A''_0$ off-path values trying to hide its malicious activity from the base station.

$$A'_0 = < A_{id}, 1, 100, H(N||1||10) >$$
$$B'_0 = < B_{id}, 1, 200, H(N||1||20) >$$

$$C'_1 = < C_{id}, 2, 300, H(N||2||300||\mathbf{A''_0}||\mathbf{B_0}) > \; or$$
$$C''_1 = < C_{id}, 2, 300, H(N||2||300||\mathbf{A_0}||\mathbf{B''_0}) >$$

$$B''_0 =< B_{id}, 1, 290, H(N||1||20) >$$
$$A''_0 =< A_{id}, 1, 280, H(N||1||10) >$$

$A$ and $B$ receives either $C_1'$ or $C_1''$ from the base station based on what $C$ has sent to base station.

$A$ and $B$ receives $B_0''$ and $A_0''$ from $C$ respectively.

$A$ and $B$ derives the root data-item using the received off-path values, and it does not match with the received root data-item.

$A$ uses $(A_0, B_0'')$ and derives $< 2, 300, H(N||2||300||\mathbf{A_0}||\mathbf{B_0''}) >= C_1'' \neq C_1'$

$B$ uses $(A_0'', B_0)$ and derives $< 2, 300, H(N||2||300||\mathbf{A_0''}||\mathbf{B_0}) >= C_1' \neq C_1''$.

The commitment field makes it nearly impossible for an adversary to tamper with the data-items while creating commitment tree and/or while distributing off-path values.

## Verification of Inclusion

Sensor node verifies that no aggregation tampering has occurred on the path between its leaf vertex and the root of its commitment tree.

It is able to do so since the off-path values provide all the necessary information to perform the data-item computation.

In the previous example, $C$ receives $B_0, A_{11}$ and $A_{20}$ from its parent node and it has $C_0$.

Then it compares the derived $A_3$ with the $A_3$ received from the base station.

If those data-items are identical then the node sends the authentication code with ACK message and if those data-items are not identical then the node sends the authentication code with NACK message.

## Collection of Authentication codes

For sensor node $I$ authentication codes are given as follows:

$$\begin{aligned} &\text{Positive} : \mathsf{MAC}_{\mathsf{sk}_\mathsf{I}}(\mathsf{N}||\mathsf{ACK}) \\ &\text{Negative} : \mathsf{MAC}_{\mathsf{sk}_\mathsf{I}}(\mathsf{N}||\mathsf{NACK}) \end{aligned} \tag{4}$$

$\mathsf{sk}_\mathsf{I}$ is the secret key of the sensor node $I$, ACK and NACK are special messages.

An internal sensor node computes the XOR of its own authentication code with all the received codes, and forwards it to its parent.

For example, the sensor node $B$, sends the following authentication code to $A$.

$$\mathsf{MAC}_{\mathsf{sk}_\mathsf{E}}(\mathsf{N}||\mathsf{ACK}) \oplus \mathsf{MAC}_{\mathsf{sk}_\mathsf{F}}(\mathsf{N}||\mathsf{ACK}) \oplus \mathsf{MAC}_{\mathsf{sk}_\mathsf{B}}(\mathsf{N}||\mathsf{ACK})$$

## Verification of Authentication codes

The base station computes the $\Delta_{ack}$ as follows :

$$\Delta_{ack} = \bigoplus_{i=1}^{n} \mathsf{MAC}_{\mathsf{sk_i}}(\mathsf{N}||\mathsf{ACK})$$

Here, the addition represents an XOR operation.

The base station compares $\Delta_{ack}$ with the received root authentication code $\Delta_{root}$.

If those two codes match then it accepts the aggregated value or else it proceeds further to find an adversary.

The base station identifies the nodes who sent their authentication codes with NACK called a complainer.

The base station can find a single complainer complainer in **linear time**.

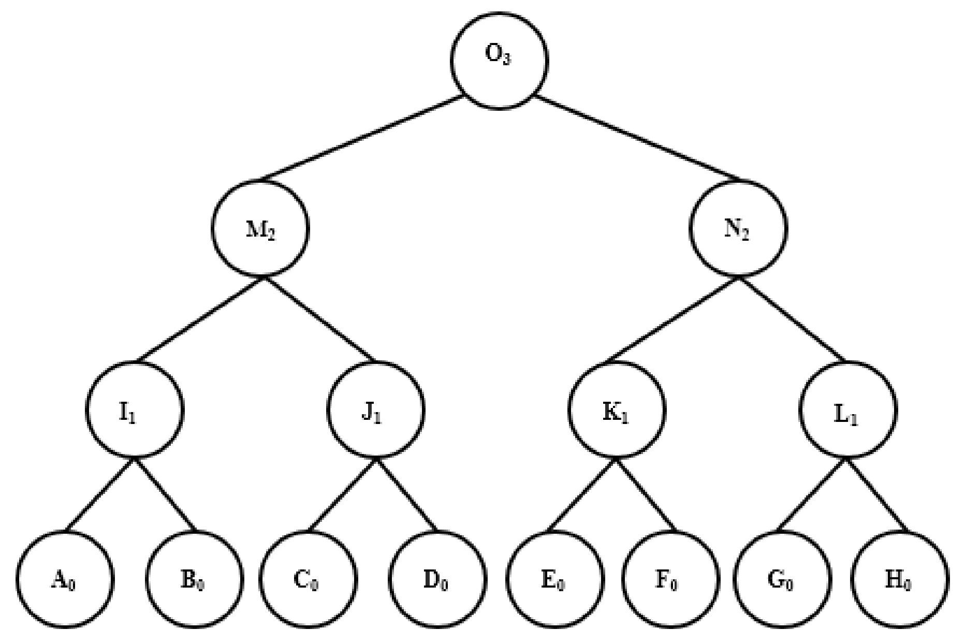
## Identifying more than one complainer

```
1: BS requests authentication codes of all the tree roots in the aggregation tree root's forest.
2: BS simulates authentication codes of all the tree roots in the aggregation tree root's forest with ACK
   message.
3: BS compares the requested and simulated authentication codes.
4: BS constructs T = $\{T_1, T_2, \ldots, T_n\}$ whose authentication codes do not match.
5: for all $t \in$ T do
6:     Call Algorithm 2 with ($t$)
```

```
1: BS requests the authentication codes of child $C_1$ and $C_2$ for the given tree.
2: BS simulates the authentication codes of child $C_1$ and $C_2$ with ACK message.
3: BS compares the requested and simulated authentication codes and identifies $C'$ whose authentication
   codes do not match.
4: if $C'$ is a leaf vertex then
5:     Add $C'$ to complainer vertex set C
6: else
7:     Call Algorithm 2 with ($C'$)
```

## Detecting an Adversary

Based on the complainer set C it constructs a set of possible adversaries A $= \{\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_n\}$.



Suppose C $= \{A_0\}$ then A $= \{I, \{B, I\}, \{B, M\}, \{B, I, M\}\}$.

Note the fact that only $I$ knows the exact data-item of $A_0$.

The vertices $M$ and $O$ knows aggregated value of $A_0$.

Hence, if either $M$ or $O$ tampers they tamper with more than one data-item or they have to cheat in a group.

## Detecting a single adversary

Suppose, $I$ is an adversary and C $= \{A_0\}$ then A $= \{I, \{B, I\}, \{B, M\}, \{B, I, M\}\}$.

$$I_1 = < I_{id}, 1, I_{1value}, H(N||1||I_{value}||\mathbf{A_0}||B_0) >$$
$$I_{pay} = < I_1, \mathsf{Sign}_{\mathsf{sk_I}}(I_1), \mathsf{Sign}_{\mathsf{sk_I}}(I_\tau) >$$

Whereas,

$$I'_1 = < I_{id}, 1, I'_{1value}, H(N||1||I'_{value}||\mathbf{A'_0}||B_0) >$$
$$I'_{pay} = < I'_1, \mathsf{Sign}_{\mathsf{sk_I}}(I'_1), \mathsf{Sign}_{\mathsf{sk_I}}(I_\tau) > \text{ where } I_\tau = I'_1.$$

First the base station asks $A$ to send its payload.

$$A_{pay} = < A_0, \mathsf{Sign}_{\mathsf{sk_A}}(A_0), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) >$$
$$A_0 = < A_{id}, 1, A_{value}, H(N||1||A_{value}) >$$

The base station asks $I$ to send its own payload. $I$ sends its false payload to the base station.

$$A_{pay} =< A_0, \mathsf{Sign}_{\mathsf{sk_A}}(A_0), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) > \text{ where } A_\tau = A_0$$
$$B_{pay} =< B_0, \mathsf{Sign}_{\mathsf{sk_B}}(B_0), \mathsf{Sign}_{\mathsf{sk_A}}(B_\tau) > \text{ where } B_\tau = B_0$$
$$I_{pay} =< I_1, \mathsf{Sign}_{\mathsf{sk_I}}(I_1), \mathsf{Sign}_{\mathsf{sk_A}}(I_\tau) > \text{ where } I_\tau = I_0$$

Then the base station asks $M$ to send all the received payload from each of its children.

$$I'_{pay} =< I'_1, \mathsf{Sign}_{\mathsf{sk_I}}(I'_1), \mathsf{Sign}_{\mathsf{sk_I}}(I_\tau) > \text{ where } I_\tau = I'_1$$
$$J_{pay} =< J_1, \mathsf{Sign}_{\mathsf{sk_I}}(J_1), \mathsf{Sign}_{\mathsf{sk_A}}(J_\tau) > \text{ where } J_\tau = J_0$$

As the base station receives the $I'_{pay}$ with a signature from $M$, it proves that $I$ is an adversary.

## Algorithm to Detect an Adversary

```
1: BS identifies all the complainer and constructs C $= \{C_1, C_2, \ldots, cC_n\}$
2: for all $C \in$ C do
3:     BS asks $C$ to send data-item with its signature, sent during commitment tree generation phase
4: BS identifies possible adversary based on C and constructs A $= \{A_1, A_2, \ldots, A_n\}$
5: for all $A \in$ A do
6:     BS asks $A$ to send data-items with its signature, received and sent by $A$ during commitment tree
       generation phase
7:     If needed BS asks the parent of $A$ to send data-items with its signature
8: BS determines the adversary based on the verification of signatures
```

Once the base station detects adversaries it dispels them from the network for all the future queries.

Doing so removes un-necessary communication for debugging purpose in the future which saves bandwidth and increases the life-time of the network.

## Binary Tree is optimal

**Theorem 1.** *Binary commitment tree is optimal in terms of verification for $m$-ary tree, as it requires minimum number of off-path values.*

*Proof.* Consider the case of a tertiary tree, other $m$-ary tree arguments follows the same manner. Let $m$ be the number of leaves in a commitment tree.

For the given binary commitment tree, each leaf vertex needs $\log_2 m$ off-path values in the verification phase. The total off-path values needed in the given commitment tree is [$m \log_2 m$]. For the given tertiary commitment tree, each leaf vertex needs $2 \log_3 m$ off-path values in the verification phase. The total off-path values needed in given commitment tree is [$2m\log_3 m$].

Let $y = \log_3 m$

$y = \frac{\log_2 m}{\log_2 3}$ where $\log_2 3 > 1$

$y\log_2 3 = \log_2 m$

$\log_3 m \cdot \log_2 3 = \log_2 m$

$\log_3 m = \frac{\log_2 m}{\log_2 3}$

$2m\log_3 m = 2m\frac{\log_2 m}{\log_2 3} = \frac{2}{\log_2 3} m\log_2 m$

$2m\log_3 m = (1.2618)m\log_2 m$

$2m\log_3 m > m\log_2 m$

Hence, in totality the binary commitment tree requires the minimum number of off-path values. $\square$

## Analysis

For an aggregation tree with $n$ nodes, at max there will be $(\mathbf{2^{\lg n+1} - 1}) = (\mathbf{2n - 1})$ vertices in the commitment tree.

In FSwRD and FSwoRD, there are at least $\mathbf{2(2n - 1)}$ signatures created while creating the commitment tree.

In general, an intermediate node with $n$ descendants receives $\lceil \log_{\mathbf{2}} \mathbf{n} \rceil$ trees from its children.

The node with $n$ descendants receives at least $\mathbf{2}\lceil \log_{\mathbf{2}} \mathbf{n} \rceil$ signatures.

We claim that the binary representation of a non-negative number $x$ illustrates the payload decomposition of the sensor node $S$, where $x$ = $1$ + number of descendants of $S$.

Suppose, sensor node $S$ has $22$ descendants then $x = 23$, $(x)_{10}$ = $(10111)_2$. This means $S$ has four complete binary trees in its payload, with the height of four, two, one and zero.

## Analysis

| Network Topology | Totality of Signatures Transmitted | |
|---|---|---|
| | FSwRD | FSwoRD |
| Star | $2n$ | $2n$ |
| Palm Tree | $2n$ | $2n$ |
| Complete Binary | $2n$ | $2n$ |

| Network Topology | Totality of Certificates | |
|---|---|---|
| | FSwRD | FSwoRD |
| Star | $n-1$ | $n-1$ |
| Palm Tree | $XX$ | $n-1$ |
| Complete Binary | $n-1$ | $n-1$ |

# Conclusion And Future Work

Improved SHIA by removing the redundant field in its data-item.

An efficient mathematical way of analyzing the protocol.

An algorithm to find all possible cheater in the network.

An algorithm which can detect an adversary in the network and remove it for all future queries.

The proof that complete binary tree is an optimal data structure to build the commitment tree.

In the future, we would like to analyze this protocol for various network topologies and generalized the results.

It is our hope that further research can improve the cheater detection algorithm.

Thank You.

Questions ?