Secure Data Aggregation Protocol for Sensor Networks

Introduction

- Ubiquitous Computing - is a scenario in which computing is omnipresent.
- 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 information an important action is taken.

Figure 1: Sensor Network

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 A330-203 AF 447 on 1st June 2009.

France's Bureau of Investigation and Analysis (BEA) reported, the pilots could not reclaim control as the plane dropped out of the sky at a rate of 10,000 feet per minute.

The findings from the flight's black boxes, and their analysis paints a harrowing picture of Air France flight 447's literal dropping out of the sky.

The co-pilots encountered trouble with the speed sensors four hours and 10 minutes into the flight.

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

By the time the pilot returned, the plane had started to fall at 10,000 feet per minute while violently rolling from side to side.

The plane's speed sensors never regained normal functionality as the plane began its three-and-a-half minute freefall.

The flight plunged into the Atlantic nose-up, killing all 228 on board.

Resource Constrains in Sensor Network

Physical Limitations - often deployed in open, hostile and unattended environments. Vulnerable to physical tampering due to the lower physical security.

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

Transmission Medium - sensors communicate over the wireless network using radio which has issues with synchronization, hidden station and expose station terminal problems, directional antennas, bandwidth limitations, higher error rate, security, scalability etcetera. For example, 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. It makes difficult to do the routing in the network. It requires the network to be agile enough to do the reconfiguration for the network topology.

Cryptographic Tools

Cryptanalysis is the science breaking of cryptography schemes. Formally, the basic component of cryptography is a cryptosystem.

Definition 0.1. *A cryptosystem is a 5-tuple $(\mathcal{E}, \mathcal{D}, \mathcal{M}, \mathcal{K}, \mathcal{C})$, where $\mathcal{M}$ is the set of plaintexts, $\mathcal{K}$ is the set of keys, $\mathcal{C}$ is the set of ciphertexts, $\mathcal{E} : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ is the set of enciphering functions, and $\mathcal{D} : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ is the set of deciphering functions.*

Symmetric Key And Asymmetric Key Encryption

Consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$, respectively, where $\mathcal{K}$ is the key space.

The encryption scheme is said to be **Symmetric-Key** if for each associated encryption/decryption key pair (e, d), it is computationally "easy" to determine d given e, and to determine e from d. Moreover, most symmetric-key schemes satisfy $e = d$.

The encryption scheme is said to be **Asymmetric-Key** if for any pair of associated encryption/decryption transformations (E_e, D_d) and assuming each pair has the property that knowing E_e it is computationally infeasible, given a random ciphertext $c \in \mathcal{C}$, to find the message $m \in \mathcal{M}$ such that $E_e(m) = c$. This property implies that given e it is infeasible to determine the corresponding decryption key d.

Hash Functions

A hash function takes a message as its input and outputs a fixed length message called hash code. The hash code represents a compact image of the message like a digital fingerprint. Hash functions are essential mathematical tools to achieve data integrity. A hash function h should have the following properties :

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

Ease of computation For given h, x it is easy to compute $h(x)$.

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, also known as keyed hash function (h_k). It has the following properties :

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

Compression The 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 cannot deny having sent the message (authentication and non-repudiation) and that the message was not altered in transit (integrity).
A Digital Signature scheme consists of the following :

1. a plain text message space $\mathcal{M}$ (set of strings over alphabets)
2. a signature space $\mathcal{S}$ (set of possible signatures)
3. a signing key space $\mathcal{K}$ (set of possible keys for signature generation) and a verification space $\mathcal{K}^{'}$ (a set of possible verification keys)
4. an efficient key generation algorithm Gen : $N \rightarrow \mathcal{K} \times \mathcal{K}^{'}$
5. an efficient signing algorithm Sign : $\mathcal{M} \times \mathcal{K} \rightarrow \mathcal{S}$
6. an efficient verification algorithm Verify : $\mathcal{S} \times \mathcal{M} \rightarrow \{$true, false$\}$

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:

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

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:

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

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.

The message is hashed before its being signed to reduce the message size. If the message is not hashed before signing then the signature can be longer than the message which is problematic for the longer messages.

Summary of Cryptography Tools

Three different integrity-protection mechanisms HASH, MAC, Signature can be summarized in a matrix like Table 1.
The main difference between the various primitives stems from identifying who can generate the code and who can verify it.

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

Table 1: A comparison of integrity-protecting primitives

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, the data will be propagated in some fashion to the base station. At the base station, the data is aggregated and processed 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.

Here f is some mapping $f : \mathcal{D}_1 \times \mathcal{D}_2 \times ...\mathcal{D}_n \to I$, where $\mathcal{D}_i$ represents sensor i's domain and I represents the set of all possible information.

Thus the goal is to compute the information as follows :

$$y = f(x_1, x_2, ..., x_n) \tag{3}$$

Data Aggregation using SUM

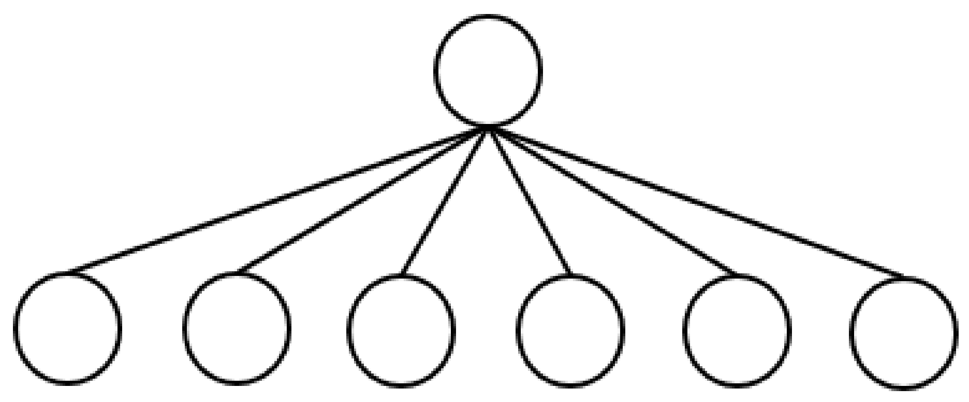

Figure 2: Star Network

The root in the star topology of Figure 2, receives the following sensor data

$S_1(10), S_2(14), S_3(12), S_4(15), S_5(11), S_6(17)$ and its own data $S_0(15)$.

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

$$S = \sum_{i=0}^{6} S_i \tag{4}$$

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

Common Topologies

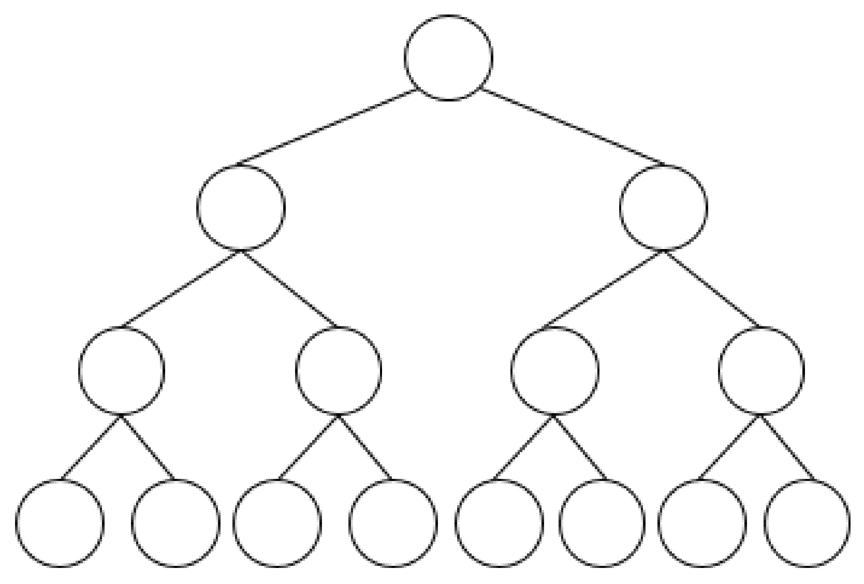

Figure 3: Binary Tree Network

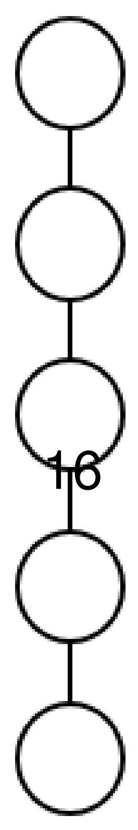

Figure 4: Palm Tree Network

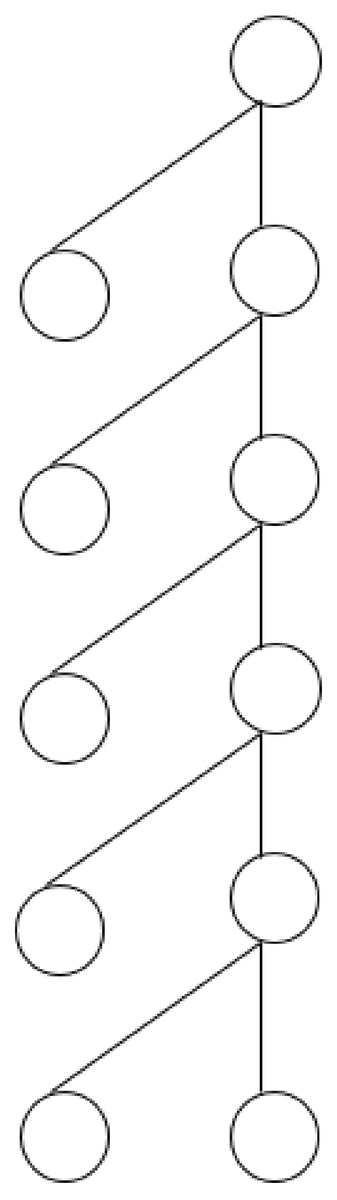

Figure 5: Palm Tree Network

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 similar to 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

Secure hierarchical in-network aggregation (SHIA) is a light wait protocol providing data-integrity to in-network data aggregation.

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

We assume a multi hop network with a set $S = \{S_1, ..., S_n\}$ of n sensor 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 has more computation, storage capacity then the sensor nodes in the network.

The base station knows total number of sensor nodes in the network and the network topology.

It also has the capacity to directly communicate with every sensor node in the network using authenticated broadcast.

All the wireless communications between the nodes are 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 $\mathsf{sk_I}$ with the base station.

All the sensor nodes are capable of doing symmetric key encryption and symmetric key decryption.

They are also capable of computing collision resistant cryptographic hash function.

Attacker Model

We consider a model with a polynomially bounded adversary, which has a complete control over some of the sensor nodes in the network.

The adversary can change the measured values reported by sensor nodes under its control. For example, an adversary could report fictitious values (probably completely independent of the measured reported values), instead of the real aggregate values.

The adversary can misbehave in any arbitrary way, and the only limitations we put on the adversary are its computational resources (polynomial in terms of the security parameter) and the fraction of nodes that it can have control over.

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), passively sniffing the traffic and releasing the message contents attacks.

SUM Aggregate Algorithm

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

SHIA requires the prior constructed rooted aggregation tree.

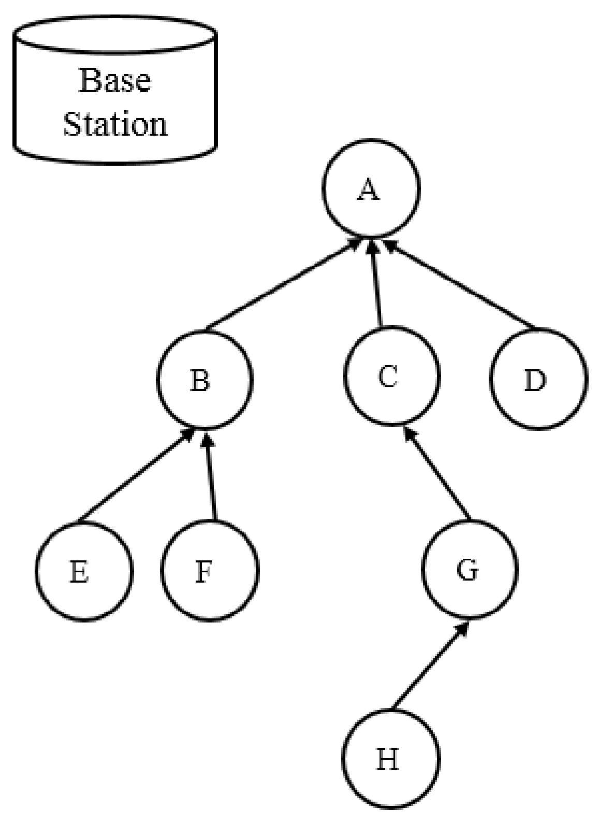

Figure 6: Aggregation Tree.

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.

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.

A hash chain prevents an adversary from predicting the query nonce for future queries as it has to reverse the hash chain computation to get an acceptable pre-image.

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 as follows :

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

Each sensor node creates its own data-item, For example, sensor node A creates its data-item A_0 as follows:

$$A_0 = < A_{id}, 1, A_{value}, H(N||1||A_{value}) > . \tag{5}$$

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.

$$\mathsf{S} = \mathsf{Sign}_{\mathsf{sk_A}}(A_0) \tag{6}$$

Table 2: Digital Certificate

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

$$\mathsf{Verify}_{\mathsf{pk_A}}(A_0, \mathsf{S}) = \begin{cases} \textbf{true} \text{ with probability of 1} & \text{if } \mathsf{S} = \mathsf{Sign}_{\mathsf{sk_A}}(A_0) \\ \textbf{false} \text{ with overwhelming probability} & \text{if } \mathsf{S} \neq \mathsf{Sign}_{\mathsf{sk_A}}(A_0) \end{cases} \tag{7}$$

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

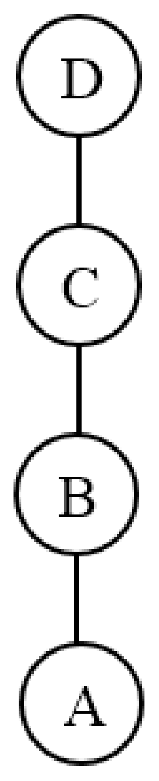

Figure 7: Palm Shaped Aggregation Tree

$$A_{pay} = < A_0, \mathsf{Sign}_{\mathsf{sk_A}}(A_0), \mathsf{Sign}_{\mathsf{sk_A}}(A_\tau) > \ where \ A_\tau = < A_0 > \quad (8)$$

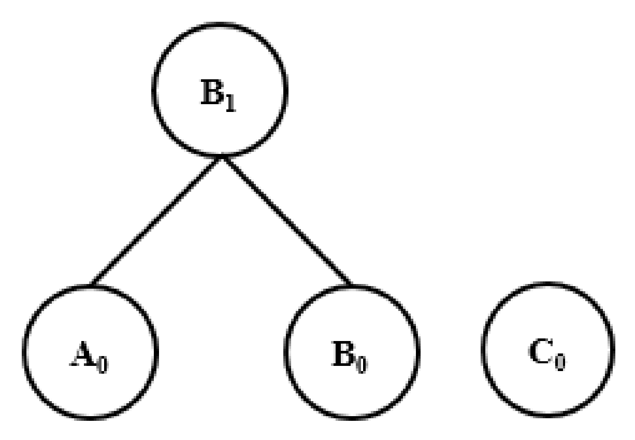

Figure 8: Commitment Payload of Sensor Node C

$$C_{pay} = < C_0, \mathsf{Sign}_{\mathsf{sk_C}}(C_0), B_1, \mathsf{Sign}_{\mathsf{sk_C}}(B_1), \mathsf{Sign}_{\mathsf{sk_C}}(C_\tau) > \; where \, C_\tau = < B_1 \,||\, C_0 > \quad (9)$$

Query Dissemination

Aggregate Commit

We describe the commitment tree generation process for an aggregation tree shown in Figure 6.

LaTeX Workshop