SECURE DATA AGGREGATION SCHEME FOR SENSOR NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kavit Shah

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Electronics Engineering

December 2014

Purdue University

Indianapolis, Indiana

This is the dedication.

ACKNOWLEDGMENTS

This is the acknowledgments.

PREFACE

This is the preface.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

m	mass
v	velocity

ABBREVIATIONS

abbr	abbreviation
bcf	billion cubic feet
BMOC	big man on campus

NOMENCLATURE

Alanine	2-Aminopropanoic acid
Valine	2-Amino-3-methylbutanoic acid

GLOSSARY

chick	female, usually young
dude	male, usually young

ABSTRACT

Shah, Kavit Master, Purdue University, December 2014. Secure data aggregation scheme for sensor networks. Major Professor: Dr. Brian King.

This is the abstract.

1. INTRODUCTION

2. SECURITY/DATA AGGREGATION BACKGROUND

Cite papers read and also summarize

[1] [2]

3. NETWORKING AND CRYPTOGRAPHY TOOLS

Networking - Algorithms of generating tree from a given graph. Optimal tree structure.

Hash

Elliptic curve

4. IN-NETWORK DATA AGGREGATION OVERVIEW

4.1 In-network data aggregation

Sensor networks are used in scientific data collection, emergency fire alarm systems, traffic monitoring, wildfire tracking, wildlife monitoring and many other applications. In sensor networks, thousands of sensor nodes may interact with the physical environment and collectively monitors an area, generating a large amount of data to be transmitted and reasoned about. The sensor nodes in the network often have limited resources, such as computation power, memory, storage, communication capacity and most significantly, battery power. Furthermore, data communication between nodes consumes a large portion of the total energy consumption. The in-network data aggregation reduces the energy consumption by eliminating redundant data being transmitted to the base station. For example, in-network data aggregation of the SUM function can be performed as follows. Each intermediate sensor node in the network forwards a single sensor reading containing the sum of all the sensor readings from all of its descendants, rather than forwarding each descendants sensor reading one at a time to the base station. It is shown that the energy savings achieved by in-network data-aggregation are significant [3]. The in-network data aggregation approach requires the sensor nodes to do more computations. But studies have shown that data transmission requires more energy than data computation. Hence, in-data aggregation is an efficient and a widely used approach for saving bandwidth by doing less communications between sensor nodes and ultimately giving longer battery life to sensor nodes in the network.

We define the following terms to help us define the goals of in-network data-aggregation approach.

Definition 4.1.1 *[4]* ***Payload*** *is the part of the transmitted data which is the fundamental purpose of the transmission, to the exclusion of information sent with it such as metadata solely to facilitate the delivery.*

Definition 4.1.2 ***Information-rate*** *for a given node is the ratio of the* ***payloads****, number of* ***payloads*** *sent divided by the number of* ***payloads*** *received.*

The goal of the aggregation process is to achieve the lowest possible ***information rate***. In the following sections, we show that reducing ***information rate*** makes the intermediate (aggregator) sensor nodes more powerful. Also, it makes aggregated ***payload*** more fragile and vulnerable to various security attacks. Now, we describe bandwidth analysis of different in-network aggregation approaches.

4.2 Bandwidth analysis

Congestion is widely used parameter while doing bandwidth analysis of networking applications. The congestion for any given node is defined as follows:

$$Congestion = edgeCongestion * fanout \tag{4.1}$$

Congestion is very useful factor while analyzing sensor network as it measures how quickly the sensor nodes will exhaust their batteries [5]. To transmit a k - bit packet at distance d, the energy dissipated is:

$$E_{tx}(k, d) = E_{elec} * k + \varepsilon_{amp} * k * d^2 \tag{4.2}$$

and to receive the k - bit packet, the radio expends

$$E_{rx}(k) = E_{elec} * k \tag{4.3}$$

For μAmp wireless sensor, $E_{elec} = 50nJ/b$ and $\varepsilon_{amp} = 100pJ/b/m^2$. cite papers and add more details on the equations

Some nodes in the sensor network have more congestion than the other. The highly congested nodes are the most important to the the network connectivity, for example, the nodes closer to the base station are essential for the network connectivity. The failure of the highly congested nodes may cause the sensor network to fail even though most of the nodes in the network are working. Hence it is desirable to have a lower congestion on the highly congested nodes even though it costs more congestion within the overall sensor network.

To achieve the lowest possible ***information rate***, we can construct an aggregation protocol where each node transmits a single **data-item** defined in $X.X$ to its parent in the aggregation tree. It implies there is $\Omega(1)$ congestion on each edge in the aggregation tree, thus resulting in $\Omega(f)$ congestion on the node, where f is the fanout of that node according to Definition 4.1. In this approach, f is dependent on the given aggregation tree, which can be $O(n)$ for the star tree topology and $O(1)$ for the palm tree topology. This can create some highly congested nodes in the aggregation tree which is highly undesirable. In most of the real world applications we cannot control f as the aggregation tree is random. Hence, it is desirable to have almost uniform ***information rate***across the aggregation tree.

Talk about No aggregation approach.

SHIA tries to achieve uniform congestion in the network.

4.3 Security in In-network data aggregation

In-network data aggregation approach saves bandwidth by transmitting less ***payloads*** between sensor nodes but it gives more power to the intermediate aggregator sensor nodes. For example, a malicious intermediate sensor node who is doing aggregation over all of its descendants ***payloads***, needs to tamper with only one aggregated ***payload*** instead of tampering with all the ***payloads*** received from all of its descendants. Thus, a malicious intermediate sensor node needs to do less work to skew the final aggregated ***payload***. An adversary controlling few sensor nodes in the network can cause the network to return unpredictable ***payloads***, making an entire sensor network unreliable. Notice that the more descendants an intermediate sensor node has the more powerful it becomes. Despite the fact that in-network aggregation makes an intermediate sensor nodes more powerful, some aggregation approaches requires strong network topology assumptions or honest behaviors from the sensor nodes. For example, in-network aggregation schemes in [5, 6] assumes that all the sensor nodes in the network are honest. Secure Information Aggregation (SIA) of [7], provides security for the network topology with a single-aggregator model.

Secure hierarchical in-network aggregation (*SHIA*) in sensor networks [8] presents the first and provably secure sensor network data aggregation protocol for general networks and multiple adversaries. We discuss the details of the protocol in the next chapter. *SHIA* limits the adversary's ability to tamper with the aggregation result with the tightest bound possible but it does not help detecting an adversary in the network. Also, we claim that same upper bound can be achieved with compact label format defined in the next chapter.

5. SECURE HIERARCHICAL IN-NETWORK DATA AGGREGATION

We describe the Secure Hierarchical In-network data aggregation *(SHIA)* protocol of [8] as our work enhances this protocol by making it more efficient and adding new capabilities to the protocol. The goal of *SHIA* is to compute aggregate functions (such as *SUM*, *AVERAGE*, *COUNT*) of the sensed values by the sensor nodes while assuming that a portion of the sensor nodes are controlled by an adversary which is attempting to skew the final result.

5.1 Network Assumptions

5.2 Security Infrastructure

5.3 Attacker Model

5.4 Problem Definition

5.5 The SUM Aggregate Algorithm

In this algorithm, the aggregate function f is addition meaning that we want to compute $a_1 + a_2 + \ldots + a_n$, where a_i is the sensed data value of the node i. This algorithm has three main phases:

- Query dissemination
- Aggregate commit
- Result checking

5.6 Query dissemination

Prior to this phase an aggregation trees is created using a tree generation algorithm. We can use any tree generation algorithm as this protocol works on any aggregation tree structure. For completeness of this protocol, one can use Tiny Aggregation Service (TaG) [3]. TaG uses broadcast message from the base station to initiate a tree generation. Each node selects its parent from whichever node it first receives the tree formation message. One possible aggregation tree for given network graph in Figure 5.1 is shown in Figure 5.2.

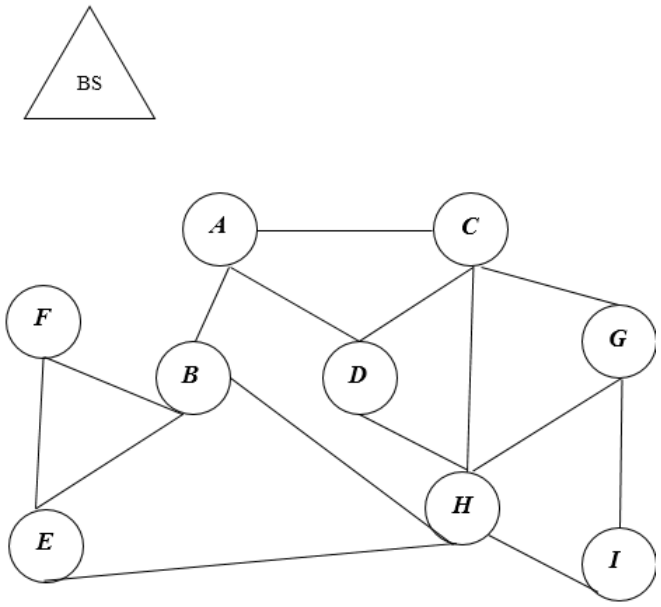

Fig. 5.1.: Network graph

To initiate the query dissemination phase, the base station broadcasts the query request message with the query nonce N in the aggregation tree. The query request message contains new query nonce N for each query to prevent replay attacks in the network. It is very important that the same nonce is never re-used by the base station. $SHIA$ 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 $h^m(R)$, then the next broadcast uses $h^{m-1}(R)$ as the nonce; R is the initial random value. 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.

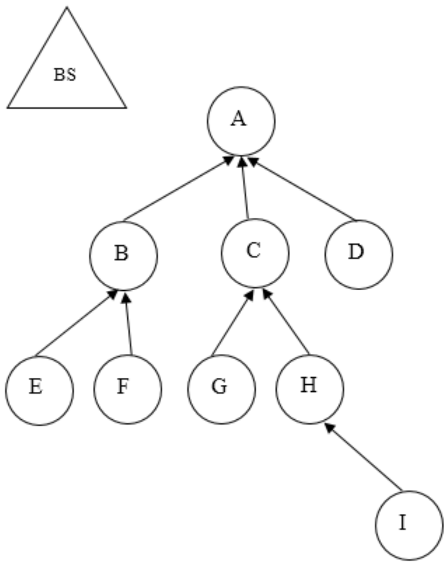

Fig. 5.2.: Aggregation tree for network graph in Figure 5.1

5.7 Aggregate commit

The aggregation-commit phase constructs cryptographic commitments to the data values and to the intermediate in-network aggregation operations. These commitments are then passed on to the base station by the root of an aggregation tree. The base station then rebroadcasts the commitments to the sensor network using an authenticated broadcast so that the rest of the sensor nodes in the network can verify that their respective data values have been incorporated into the final aggregate value.

5.7.1 Aggregate commit: Naive Approach

In the naive approach, during aggregation process each sensor node computes a cryptographic hash of all its inputs (including its own data value). The aggregation result along with the hash value called a label, is then passed on to the parent in the aggregation tree. The label is defined in Definition **??** Figure 5.3 shows a commitment tree. Conceptually, a commitment tree is a hash tree defined in [7] with additional aggregate information attached to the nodes to help us in result checking phase.

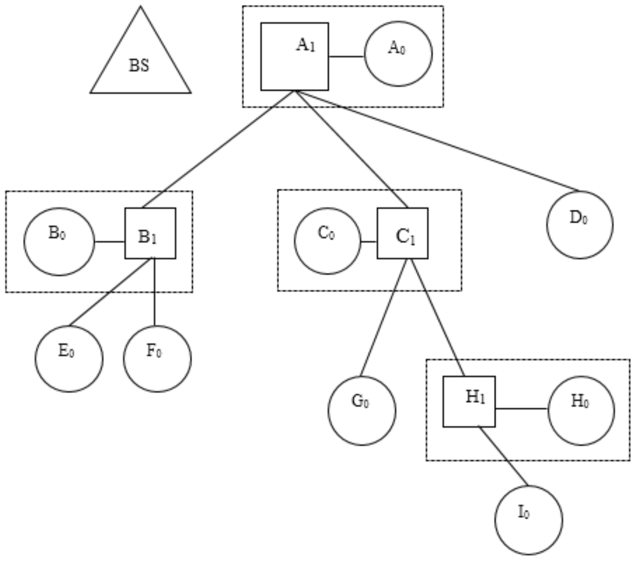

Fig. 5.3.: Naive commitment tree

Definition 5.7.1 *Definition 3 A commitment tree is a tree where each vertex has an associated label representing the data that is passed on to its parent. The labels have the following format: count, value, complement, commitment Where count is the number of leaf vertices in the subtree rooted at this vertex; value is the SUM aggregate computed over all the leaves in the subtree; complement is the aggregate over the COMPLEMENT of the data values; and commitment is a cryptographic commitment.*

Then elaborate your approach. Two differences: data-item format CT generation being root in as many trees as possible

6. A PROTOCOL FOR COMMITMENT TREE GENERATION

7. VERIFICATION

7.1 dissemination final commitment

7.2 dissemination of off-path values

7.3 verification of inclusion

7.4 collection of authentication codes

7.5 verification of authentication codes

The authentication codes for sensor node s, with either positive or negative acknowledgment message, are defined as follows:

$$MAC_{K_s}(N \mid\mid ACK) \tag{7.1}$$

$$MAC_{K_s}(N \mid\mid NACK) \tag{7.2}$$

K_s is the key that s shares with the base station; ACK, $NACK$ are special messages for positive and negative acknowledgment respectively. The authentication code with ACK message is sent by the sensor node if it verifies its contribution correctly to the root commitment value during the *verification of inclusion* phase and vice versa.

To verify that every sensor node has sent its authentication code with ACK, the base station computes the Δ_{ack} as follows:

$$\Delta_{ack} = \bigoplus_{i=1}^{n} MAC_{K_i}(N||ACK) \tag{7.3}$$

The base station can compute Δ_{ack} as it knows K_s for each sensor node s. Then it compares the computed Δ_{ack} with the received root authentication code Δ_{root} from the root of the aggregation tree. If those two codes match then it accepts the aggregated value or else it proceeds further to find an adversary.

To detect an adversary, the base station needs to identify which nodes in the aggregation tree sent its authentication codes with $NACK$ during the verification of inclusion phase. The node who sent authentication code with $NACK$ during the verification of inclusion phase is called a *complainer*. We claim that if there is a single complainer in the aggregation tree during the verification of inclusion phase then the base station can find the complainer in linear time. To find a complainer, the base station computes the complainer code c according to Equation 7.4.

$$c = \Delta_{root} \oplus \Delta_{ack} \tag{7.4}$$

Then it computes the complainer code c_i for node i according to Equation 7.5.
$\forall i \in [1, n]$

$$c_i = MAC_{K_i}(N \mid\mid ACK) \oplus MAC_{K_i}(N \mid\mid NACK) \tag{7.5}$$

Then it compares c with all c_i one at a time. The matching code indicates the complainer node. The base station needs to do n comparison to find a complainer in the aggregation tree. Hence, the base station can find a single complainer in linear time. For example, if there are four nodes s_1, s_2, s_3, s_4 in the aggregation tree and their authentication codes with ACK, $NACK$ messages in the binary format are defined below. The base station receives $\Delta_{root} = (0100)_2$.

$$MAC_{K_1}(N \mid\mid ACK) = (1001)_2 \ ; MAC_{K_1}(N \mid\mid NACK) = (1101)_2 \tag{7.6}$$

$$MAC_{K_2}(N \mid\mid ACK) = (0110)_2 \ ; MAC_{K_2}(N \mid\mid NACK) = (1111)_2 \tag{7.7}$$

$$MAC_{K_3}(N \mid\mid ACK) = (0101)_2 \ ; MAC_{K_3}(N \mid\mid NACK) = (0111)_2 \tag{7.8}$$

$$MAC_{K_4}(N \mid\mid ACK) = (0011)_2 \ ; MAC_{K_4}(N \mid\mid NACK) = (1110)_2 \tag{7.9}$$

Then according to Equation **??**, $\Delta_{ack} = (1101)_2$
And according to Equation 7.5, $c_1 = (0100)_2$, $c_2 = (1001)_2$, $c_3 = (0010)_2$, $c_4 = (1101)_2$
And according to Equation 7.4, $c = (1101)_2$ So, the base station identifies that the s_4 complained, during verification of inclusion phase.

for more than one complainer it becomes exponential.

How XOR is negating the contribution of NACK.

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 1 \end{pmatrix}$$

The base station receives the following:

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 \end{pmatrix}$$

The base station does the following:

$$\left(\begin{array}{cccc|cccc|cccc|cccc} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}\right)$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 1 \end{pmatrix}$$

And concludes that node 4 is complaining.

7.6 Detect an adversary

Algorithm 1 Pseudo algorithm to detect an adversary

```
1: $BS$ identifies all the complainer and creates $c = \{c_1, c_2, \ldots, c_n\}$
2: for all $N \in c$ do
3:     $BS$ asks $N$ to send data-items with its signature, sent during commitment tree
       generation phase
4: $BS$ identifies possible adversary based on $c$ and creates $a = \{a_1, a_2, \ldots, a_n\}$
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 $A's$ parent to send data-items with its signature
8: $BS$ determines the adversary
```

LIST OF REFERENCES

LIST OF REFERENCES

[1] A. Wang, W. B. Heinzelman, A. Sinha, and A. P. Chandrakasan, "Energy-scalable protocols for battery-operated microsensor networks," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 29, no. 3, pp. 223–237, 2001.

[2] M. Ettus, "System capacity, latency, and power consumption in multihop-routed ss-cdma wireless networks," in *Radio and Wireless Conference, 1998. RAWCON 98. 1998 IEEE.* IEEE, 1998, pp. 55–58.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[4] Payload computing. [Online]. Available: http://en.wikipedia.org/wiki/Payload_(computing)

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data.* ACM, 2003, pp. 491–502.

[6] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *ACM Sigmod Record*, vol. 31, no. 3, pp. 9–18, 2002.

[7] B. Przydatek, D. Song, and A. Perrig, "Sia: Secure information aggregation in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems.* ACM, 2003, pp. 255–265.

[8] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proceedings of the 13th ACM conference on Computer and communications security.* ACM, 2006, pp. 278–287.