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

Page

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

TIPS: USE ACTIVE VOICE USE VERBS DON'T TURN VERBS INTO NOUNS

COMMON MISTAKE: DATA ARE ; DATA IS PLURAL THAT/WHICH

Advancements in compute, storage, networks and sensors technologies have led to many new promising applications.

1.1 Sensor Networks

The sensor networks of the near future are envisioned to consist of hundreds to thousands of inexpensive wireless sensor nodes, each with some computational power and sensing capability, operating in an unattended mode. They are intended for a broad range of environmental sensing applications from vehicle tracking to habitat monitoring. Give an example and talk about energy, security constraints.

1.2 Internet Of Things

In the world of mass connectivity people need to get information all the time on an array of devices. Everything from your refrigerator to your thermostat is connected to wireless networks and joining the "internet of things". Write about bandwidth constraints.

1.3 Big Data

All the large internet companies process massive amounts of data also know as "Big Data" in real time applications. These include batch-oriented jobs such as data mining, building search indices, log collection, log analysis, real time stream processing, web search and advertisement selection on big data. To achieve high scalability, these applications distributes large input data set over many servers. Each server process its share of the data, and generates local intermediate. The set of intermediate results contained on all the servers is then aggregated to generate the final result. Often the intermediate data is large so it is divided across multiple servers which perform aggregation on a subset of the data to generate the final result. If there are N servers in the cluster, then using all N servers to perform the aggregation provides the highest parallelism. Talk about compute constraints. [**?**]

Airplanes are also a great example of "big data". In a new Boeing Co.747, almost every part of the plane is connected to the Internet, recording and sometimes sending continuous streams of data about its status. According to General Electric Co. in a single flight one of its jet engines generates half a tera bytes of data. This shows that we have too much of data and we are just getting started.

1.4 Data Aggregation

Data aggregation is an important technique used in many system architectures. The key idea is to combine the data coming from different sources eliminating the data redundancy, minimizing the number of packet transmissions thus saving energy, bandwidth and memory usage. This technique allows us to focus more on data centric approaches for networking rather than address centric approaches. [1]

1.5 Cloud Computing

1.6 Fog Computing

2. RELATED WORK

2.1 Secure Aggreagation

David Wagner in Resilient Aggregation in Sensor Networks describes various attacks on aggregation schemes and introduces stastical estimation theory to secure aggregation. It helps deciding secure aggregation function by defining function is resilient or not.

SIA: Secure Information Aggregation in Sensor Networks proposes secure aggragation scheme for single-aggregator model. It provides stastical security properties.

3. PROBLEM STATMENT AND ASSUMPTIONS

3.1 Assumptions

1. Trusted Querier 2. Leaves can report only one value at a time. 3. pre-knowledge of network topology

4. SECURE DATA AGGREGATION SCHEME

The goal of this thesis is to examine secure data aggregation schemes for various distributed systems.

Many modern world system designs are distributed in nature. The system design includes small, individual components doing their tasks precisely and lots of these components synchronize with all other components to complete the bigger task.

Many applications of sensor network are inharenlty distributed in nature. For example, scientific data collection, building health monitoring , building safety monitoring systems are distributed systems. Write an example how data aggregation happes in one particular application. [2]

The application design architecture for the internet of things is distributed as well. Write an example how data aggregation happens in one particular application. [**?**]

4.1 Network topology

Write about how all theses distributed systems can be classifieds into general tree structure.

Subsubsection heading

This is a sentence. This is a sentence.

5. COMMITMENT TREE GENERATION

Theorem 5.0.1 *Binary commitment tree is optimal for terms of verification as it requires minimum number of off-path values.*

Proof Let us say

$\log_3(n) = y$

$3^y = n$

$\log_2(3^y) = \log_2(n)$

$y * \log_2(3) = \log_2(n)$

$\log_3(n) * \log_2(3) = \log_2(n)$

$\log_3(n) = \frac{\log_2(n)}{\log_2(3)}$

$2 * \log_3(n) = [2/\log_2(3)] * log_2(n) = (1.2618) * log_2(n)$

$2 * log_3(n) > log_2(n)$

■

6. VERIFICATION OF CONFIRMATIONS

6.1 Network Model

We assume a multihop network with a set $S = \{s1, ..., s_n\}$ of n sensor nodes. The network is organized in a tree topology, with the base station as the root of the tree. The trusted querier resides outside of the network & has more computation, storage capacity then the sensor nodes in the network. The base station and the querier knows total number of sensor nodes n and the network topology. All the wireless comuunication is peer-to-peer and we do not consider local wireless broadcast. We also assume that the querier has the capacity to do peer to peer communication with every sensor node in the network.

6.1.1 Sensor Nodes

We assume that each sensor node has a unique identifier s and shares a unique secret symmetric key K_s with the querier. We assume 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 H. We also assume that if the sensor node has to send multiple messages to its parent it can combine those messages into single packet in a way that parent node can distinguish both the messages.

6.1.2 Commitment trees

The aggregation tree is a physical network over which all the communication happens while the commitment tree is a logical network on top of aggregation tree.

In the similar manner, vertices in a commitment tree is a logical element in a graph while a sensor node is a physical device.

The commitment trees have the following properties. At most there will be $\lceil lgn \rceil$ commitment trees in the forest for an aggregation tree with n sensor nodes. The binary representation of the number of sensor nodes n in the network indicates the number of commitment trees in the forest. It also indicates the height of all the commitment trees in the forest. For instance, if there are $11_{10} = 1011_2$ sensor nodes in the aggregation tree then there will be three commitment trees in the forest. From those three commitment trees, one will be of height three, one will be of height one and one will be of height zero. At most there will be $n + (n - 1)$ vertices in the forest of commitment trees for an aggregation tree with n sensor nodes.

It is also possible that only one sensor node s in the aggregation tree, will be the root vertex of all the commitment trees in the forest.

It is also poosible that all the internal vetices in the commitment tree are of different sensor nodes in the aggregation tree.

6.1.3 Collections of confirmations

After each sensor node s has successfully performed the verification step for its leaf vertex u_s, it sends an authentication code to its parent in the aggregation tree. The authentication code for sensor node s is $\mathrm{MAC}_{K_s}(N||ACK)$ where ACK is an acknowledgemnet message, N is the query nounce and K_s is the secret key that sensor node s shares with the trusted querier. Once an internal sensor node s in the aggregation tree has received the authentication codes from all of its descendants, it computes the XOR of its own authentication code with all the received authentication codes, and forwards the result to its parent. Finally, the querier will receive a single authentication code from the base station that consists of the XOR of all the authentication codes received in the network.

6.1.4 Verification of confirmations

Since the querier knows the secret key K_s for each sensor node s and it also knows the topology of the commitment trees in the forest, it can simulate the commitment trees in the forest. The querier simulates the commitment trees in the forest by computing the following authentication codes

$\text{MAC}_{K_1}(N||ACK) \oplus \text{MAC}_{K2}(N||ACK) \oplus \ ... \ \oplus \text{MAC}_{K_n}(N||ACK)$;

$\text{MAC}_{K_1}(N||NACK) \oplus \text{MAC}_{K2}(N||NACK) \oplus \ ... \ \oplus \text{MAC}_{K_n}(N||NACK)$

and creates two simulated commitment trees, one with the authentication codes of ACK messages and one with the authentication codes of NACK messages; where ACK is an acknowledgemnet message, NACK is a negative acknowledgemnet message & N is the query nounce. Then the querier merges all the commitment trees in the forest simulated with ACK messages, by taking XOR of the root of all the commitment trees in the forest and calculates a single root authentication code for the forest simulated with ACK messages. The querier does the same procedure for the simulated commitment trees' forest with NACK messages. The querier stores all the simulated commitment trees and root authentication codes in the memory.

The querier receives a signle root authentication code from the base station in the collection of confirmation messages step. The querier compares the received root authentication code with its simulated root authentication code of ACK message commitment trees' forest. If those two values match it means every node in the network sent ACK message during collection of confirmation step. If those two values do not match it means one or more nodes in the network sent NACK message during collection of confirmation step. In the case where root authentication codes do not match, the querier will proceed futher to find out which node or nodes in the network sent NACK message.

To find out which node or nodes in the network reported NACK message, the querier will ask the base station to send the authentication codes of all the commitment trees' root in the forest. After receiving the authentication codes from the base station, the querier will compare those authentication codes with its simulated authentication codes of ACK trees. After this comparision, whose authentication codes do not match, the querier will classify those trees as BAD TREES in the commitment forest.

Then the querier will compare the authentication codes of those BAD TREES with its simulated authentication codes for NACK trees. If any of those authentication codes match it means all the nodes in that commitment tree sent NACK message during collection of confirmation stpe. If those values do not match then the querier will proceed furhter to find the senosr node or nodes who reported NACK message in BAD TREES.

6.1.5 Algorithm for detecting a node who reported NACK message

For each BAD TREE in the commitment forest it will does the following :

1. The querier compares the authentication codes of the root vertices of the BAD TREES with relavant root vertices of the simulated commitment trees with NACK messages. If those codes match it means all the vertices rooted in that subtree reported NACK messages during collection of confirmation steps. And the querier will stop doing queries to that subtree. If those codes do not match it will go to step 2.

2. The querier asks the relevant nodes in the aggregation tree to send the authentication codes of their children in the commitment trees.

3. The querier compares those codes with relavant vertices of the simulated commitment trees with ACK messages. The querier classifies the subtree rooted at those vertices as BAD SUBTREE, whose authentication codes do not match.

4. The querier compares those authentication codes of the root of the BAD SUBTREE with relavant root vertices of the simulated commitment trees with NACK messages. If those codes match it means all the vertices rooted in that

subtree reported NACK messages during collection of confirmation steps. And the querier will stop doing queries to that subtree. If those codes do not match then go to step 2.

To find the sensor node or nodes who sent NACK messages in the BAD TREES, the querier will ask the relavant sensor nodes in the aggregation tree to send the authentication codes of theier children in the commitment tree. The querier will compare those authentication codes with relevant simulated commitment trees in the forest. First the querier will compare those authentication codes with simulated trees with NACK messages. If any of those childerns' authentication codes match with the authentication codes from simulated trees of NACK messages, it means all the relevant nodes of those vertices rooted in that subtree sent NACK message. If those authentication codes do not match then the querier will compare those authentication codes with simulated trees of ACK messages. The querier will look for the node or nodes whose authentication codes do not match and then the querier will add those subtree rooted at those nodes as BAD SUBTREE. Then the querier will repeat this process recursively untill it finds the node or nodes who reported NACK messages.

6.1.6 Analysis

To simulate the commitment trees in the forest the querier has to do the following calculations. Suppose there are n sensor nodes in the network, means there will be at most $\lceil lgn \rceil$ commitment trees in the forest. If there are $n^{'}$ leaves in each commitment tree in the forest then the height($h^{'}$) of each commitment tree will be $lg(n^{'})$ and the number of intermediate vertices ($i^{'}$) in the network will be $(2^{h^{'}} - 1)$. So, there will be total of $(n^{'} + i^{'})$ * $\lceil lgn \rceil$ vertices in the forest. As the querier needs to simulate two commitment trees one for ACK messages and one for NACK messages it has to do ($2 * (n^{'} + i^{'}) * \lceil lgn \rceil$) calculations.

As we know from the properties of commitment trees, at most there will be $n + (n-1)$ vertices in the forest of commitment trees for an aggregation tree with n sensor

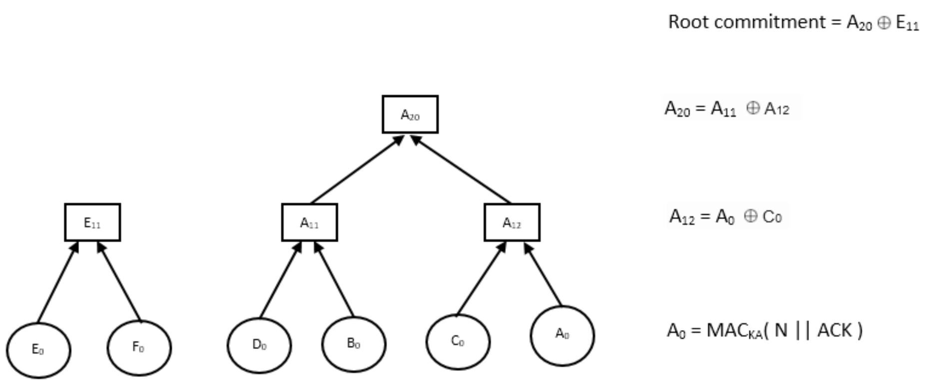

Fig. 6.1. Simulated commitment tree with ACK messages

nodes. So, the querier has to do $2 * (n + (n - 1))$ calculations and it also need that much memory space to store those values.

With Dr.King with a single NACK message in a tree:

Maximum number of leaves in a single commitment tree is equal to n. So, we can upper bound it with $O(n)$. As there is a signle NACK message in the tree, during verification phase we know in which part that vertex is.

So, we can identify the node in at most $\log n$ steps. We need $2 * \log n$ messages to identify that. The querir will have to ask $\log n$ times and some node from the aggregation time has to reply $\log n$ times. This is total number of messages is required but number of communication depends on the aggregation tree.

Worst case: In the worst case, the querier has to ask values of all the vertices in the commitment tree. At max there will be $(2 * n - 1)$ messages. Again, remember these are the number of messages, number of communications might be different.

Best case in terms of number of communications: It needs minimum number of communications when one single node is everywhere in the aggregation tree. In that case if the querier has to know the values of all the leaves vertices it has to ask $\log(n)$ questions and node has to reply to those messages. So, number of communication is equal to $2 * log(n)$.

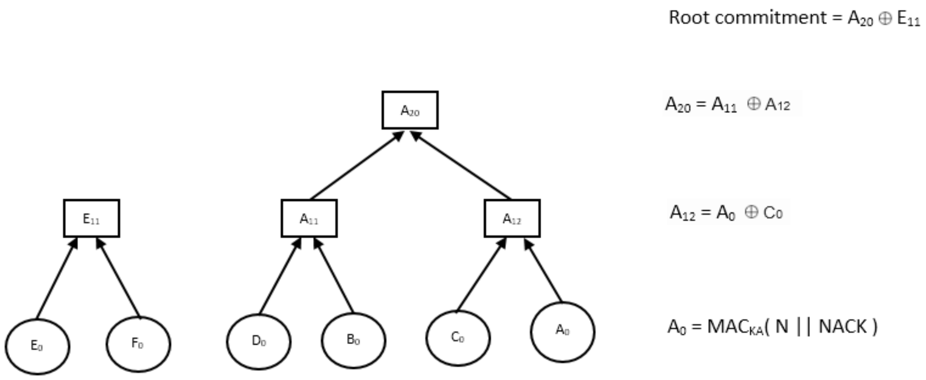

Fig. 6.2. Simulated commitment tree with NACK messages

7. AUGUST

Things discussed in meeting:

Analyzed congestion and why is it sub linear ?

In SHIA leaves verify their values with final results not with intermediate results. But in surveillance application data is compared with some base value in such network intermediate values are important.

Analyze the protocol with Digital signatures. How many signatures do we need ?

Analyze properties of commitment tree.

Definitions

A **direct data injection attack** occurs when an attacker modifies the data readings reported by the nodes under its direct control, under the constraint that only legal readings in [0, r] are reported.

An aggregation algorithm is **optimally secure** if, by tampering with the aggregation process, an adversary is unable to induce the querier to accept any aggregation result which is not already achievable by direct data injection.

For example, if A is an aggregator and it receives one reading from B. So, A needs to aggregate two values one of its own and the other is B's value. Suppose, maximum allowed value is 40. A0 = 10, B0 = 20. A1 = 30. A1 ¡= 80. If A reports any value out of that range it will get caught and any cheating within the range falls under direct data injection attack.

Congestion

As a metric for communication overhead, we consider node congestion, which is the worst case communication load on any single sensor node during the algorithm. Congestion is a commonly used metric in ad-hoc networks since it measures how quickly the heaviest-loaded nodes will exhaust their batteries [6, 12]. Since the heaviest-loaded nodes are typically the nodes which are most essential to the connectivity of the network (e.g., the nodes closest to the base station), their failure may cause the network to partition even though other sensor nodes in the network may still have high battery levels. A lower communication load on the heaviest-loaded nodes is thus desirable even if the trade-off is a larger amount of communication in the network as a whole.

For a lower bound on congestion, consider an unsecured aggregation protocol where each node sends just a single message to its parent in the aggregation tree. This is the minimum number of messages that ensures that each sensor node contributes to the aggregation result. There is $\Omega(1)$ congestion on each edge on the aggregation tree, thus resulting in $\Omega(d)$ congestion on the node(s) with highest degree d in the aggregation tree. The parameter d is dependent on the shape of the given aggregation tree and can be as large as $\Theta(n)$ for a single-aggregator topology or as small as $\Theta(1)$ for a balanced aggregation tree. Since we are taking the aggregation tree topology as an input, we have no control over d. Hence, it is often more informative to consider per-edge congestion, which can be independent of the structure of the aggregation tree.

Consider the simplest solution where we omit aggregation altogether and simply send all data values (encrypted and authenticated) directly to the base station, which then forwards it to the querier. This provides perfect data integrity, but induces O(n) congestion at the nodes and edges nearest the base station. For an algorithm to be practical, it must cause only sublinear edge congestion.

Our goal is to design an optimally secure aggregation algorithm with only sublinear edge congestion.

1. remove complement 2. variable range

8. SUMMARY

This is the summary chapter.

9. RECOMMENDATIONS

Buy low. Sell high.

LIST OF REFERENCES

LIST OF REFERENCES

[1] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on.* IEEE, 2002, pp. 575–578.

[2] D. Wagner, "Resilient aggregation in sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks.* ACM, 2004, pp. 78–87.

[3] H. Alzaid, E. Foo, and J. G. Nieto, "Secure data aggregation in wireless sensor network: a survey," in *Proceedings of the sixth Australasian conference on Information security-Volume 81.* Australian Computer Society, Inc., 2008, pp. 93–105.