ABSTRACT

Title of dissertation: Secure and Private Data Aggregation in Wireless Sensor Networks

Gelareh Taban, Doctor of Philosophy, 2008

Dissertation directed by: Professor Virgil D. Gligor
Department of Electrical and Computer Engineering

Data aggregation is an important efficiency mechanism for large scale, resource constrained networks such as wireless sensor networks (WSN). Security and privacy are central for many data aggregation applications: (1) entities make decisions based on the results of the data aggregation, so the entities need to be assured that the aggregation process and in particular the aggregate data they receive has not been corrupted (i.e., verify the integrity of the aggregation); (2) If the aggregation application has been attacked, then the attack must be handled efficiently; (3) the privacy requirements of the sensor network must be preserved.

The nature of both wireless sensor networks and data aggregation make it particularly challenging to provide the desired security and privacy requirements: (1) sensors in WSN can be easily compromised and subsequently corrupted by an adversary since they are unmonitored and have little physical security; (2) a malicious aggregator node at the root of an aggregation subtree can corrupt not just its own value but also that of all the nodes in its entire aggregation subtree; (3) since sensors have limited resourced, it is crucial to achieve the security objectives while adopting only cheap symmetric-key based operations and minimizing communication cost.

In this thesis, we first address the problem of efficient handling of adversarial attacks on data aggregation applications in WSN. We propose and analyze a detection and identification solution, presenting a precise cost-based characterization when in-network data aggregation retains its assumed benefits under persistent attacks. Second, we address the issue of data privacy in WSN in the context of data aggregation. We introduce and analyze the problem of privacy-preserving integrity-assured data aggregation (PIA) and show that there is an inherent tension between preservation of data privacy and secure data aggregation. Additionally, we look at the problem of PIA in publish-subscribe networks when there are multiple, collaborative yet competing subscribers.

Secure and Private Data Aggregation in Wireless Sensor Networks

by

Gelareh Taban

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Dedication

To **Bahareh** and **Hossein Tabatabaei**,

I am extraordinarily privileged to be your sister.

You were . . .

. . . born to catch dragons in their dens

And pick flowers

To tell tales and laugh away the morning

To drift and dream like a lazy stream

And walk barefoot across sunshine days.

James Kavanaugh "Sunshine Days and Foggy Nights"

Acknowledgments

This dissertation has benefitted tremendously from the discussion, wisdom and support of many individuals.

First and foremost, I am deeply indebted to my thesis advisor Professor Virgil Gligor for his support and advice. Professor Gligor's mentoring and insight has helped shape my outlook and approach to research and beyond. I have been very fortunate to have had the opportunity to work closely with him.

I have also had the great fortune to benefit from the mentoring and advice of Professor Jonathon Katz. His classes and advice have guided me in my studies of cryptography and provable security.

I wish to thank my friends and colleagues —Rakesh Bobba, Dr. Omer Horvitz, Dr. Himanshu Khurana, Radostina Koleva, Soo Bum Lee, Marci Meingast, Dr. Kazuhiro Minami, and Ji Sun Shin—who have served as peer mentors and collaborators throughout my studies.

There are numerous other friends, faculty and staff who have made my graduate experience one that I will cherish forever. Thank you.

I consider myself extraordinarily privileged to have as my mother, friend, advisor and colleague, Dr. Reihaneh Safavi-Naini; and as my husband, friend and colleague Dr. Alvaro A. Cárdenas. They provided amazing and unwavering support, love, patience and encouragement throughout this often-stressful and eventful period in my life. You have been an inspiration to me and aspired me to dream big and achieve bigger. I am also deeply grateful to my uncle and friend, Dr. Hamid

Safavi for his endless love and support.

During my graduate studies I have been generously supported by US Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-06-3-0001 and by the US Army Research Office under Contract W911NF-07-1-0287 at the University of Maryland. Their support has made this work possible, and I am grateful for their generosity.

Table of Contents

List of Figures

Chapter 1

Introduction

Data aggregation is an important primitive in wireless sensor networks (WSN). Data aggregation is a process that collects data from different sources and expresses the data, based on specific variables, in a summarized format. By eliminating redundant or unnecessary information from transmitted data streams, data aggregation can drastically improve the communication efficiency of a sensor network. This is especially desirable in resource-constrained networks, such as WSN, where it has been shown that radio energy dominates total energy expenditure on a sensor [RSPS02].

A significant risk of data aggregation however is that a node that is captured by an adversary can report arbitrary values as its aggregation result, thereby corrupting not only its own measurements but also that of all the nodes in its entire aggregation sub-tree. As a consequence, an adversary who captures nodes selectively and strategically (e.g., close to the BS, refer to Figure 1.1), can corrupt the entire network aggregation process, while incurring minimal cost and effort. This is called the *aggregation integrity problem.*

In this work, we focus on two issues related to the aggregation integrity problem.

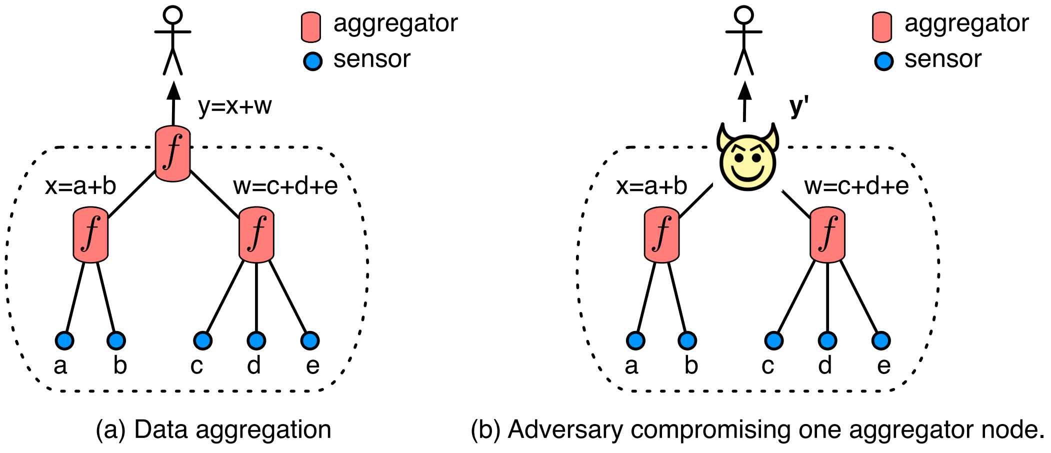


Figure 1.1: Adversary attacking data aggregation (where aggregation function is SUM). By compromising one node, the adversary can control the data of the other nodes in the network.

1. **Reliable Data Aggregation:** How can we efficiently handle integrity attacks in a data aggregation application? This includes not just detection of an attack, but also appropriate response to the attack. In particular, does in-network data aggregation retain its assumed efficiency benefits when it is applied in an adversarial setting?

2. **Data Privacy:** In many sensor applications, it is crucial that the privacy of the sensed values is preserved with respect to querying entities that receive the network data aggregate. How can we preserve the privacy of sensor data from the querying entities, while said entities also verify the integrity of the aggregation process? What are the tradeoffs between privacy and security in data aggregation?

This thesis presents advances on both the above issues. Towards reliable data aggregation, we provide a scheme where an attack against the integrity of data aggregation is not only detected but also identified. We analyze the scheme and present a cost-based characterization of the scheme that shows when data aggregation no longer provides its assumed efficiency benefits. Towards preserving data privacy in integrity-assured data aggregation, we first define and model the problem and present several classes of solutions. Our analysis of the solutions shows that there exists an intrinsic tradeoff between data privacy and integrity-assured data aggregation.

Additionally, we consider the problem of privacy-preserving integrity-assured data aggregation in publish-subscribe networks (defined in Section 1.3) which share many characteristics with WSN. In particular, we consider the scenario where there are multiple competing subscribers who want to determine the integrity of an aggregate result. We present a collaborative solution where privacy-preserving integrity-assured data aggregation can be achieved *fairly* across all subscribers. In the rest of this chapter, we provide an informal, comprehensive account of our contributions.

1.1 Reliable Data Aggregation

To achieve reliable data aggregation, and, in particular, to assure the integrity of aggregation process, it is important (*i*) to detect an adversary's presence in the network (i.e., by discovering aggregated-data corruption) and (*ii*) to identify and remove (i.e., revoke [CGPM05]) the captured nodes which corrupt data aggregates.

Most recent work on secure data aggregation has focused exclusively on efficient *detection* of integrity breach in the aggregation process; e.g., [HE03, CPS06, YWZC06, PSP03, FD08]. While detection of integrity breach is the first necessary step to achieving secure data aggregation, it does not provide a fully adequate response to malicious-node behavior; i.e., detection of integrity breach alone does not unambiguously identify and remove specific malicious nodes from the network. Exclusive reliance on detection of corrupt aggregate results would leave the network unprotected against repeated attacks that deny service to the BS. An effective approach to handling this problem would (*i*) identify corrupted nodes and remove them from the aggregation tree (e.g., by node revocation), and (*ii*) ensure continued, but gracefully degraded aggregation services, even during an attack period. Identification and removal of corrupted nodes has the added benefit of acting as a deterrent against some potential adversaries who might avoid the risk of being identified.

Problem. We consider an aggregation scenario where a subset of nodes is corrupted by an adversary. A corrupted node can (*i*) insert a false data into the network or (*ii*) if it is an aggregating node, output a false aggregation result. The goal of the corrupted node is to convince the base station to accept an invalid value. Since the network cannot protect against the insertion of incorrect aggregation values without assuming specific distributions on the environmental data [Wag04, YWZC06], we simply assume that all valid sensor inputs r must be within a given range $r_1 < r < r_2$. Our objective is to (*i*) detect an attack in the network, (*ii*) identify malicious nodes, (*iii*) ensure graceful degradation of the aggregate with respect to the number of corrupted nodes in the network, while retaining the *efficiency advantages* of data aggregation.

A straight-forward method of achieving the first three stated objectives *without* retaining in-network aggregation, henceforth called the *baseline scheme*, would be to detect the presence of malicious behavior in the network [FD08, CPS06], and then require each node to directly transmit their data *without* aggregation along with a message authentication code (MAC) to the BS. By eliminating in-network aggregation, we would trivially remove any attacks on the aggregation process. The BS can then identify any malicious nodes that inject false data by range testing the received data. If the corrupted nodes are persistently malicious, the BS can identify *all* corrupted nodes. Furthermore, the BS itself could reconstruct the network aggregate by disregarding the data received from all malicious nodes and finally guarantee the correctness of the reconstructed aggregate based on the security of the MAC protocol and data-validity verification. Although the baseline scheme would satisfy the first three objectives mentioned above, it would do so at the cost of removing in-network data aggregation and its associated communication efficiency. For this reason, we do *not* consider the baseline scheme to be a useful solution. Nevertheless, it constitutes a practical lower bound on the performance of any secure aggregation solution satisfying our three objectives above. That is, an efficient solution must have better performance than the baseline scheme; otherwise, the baseline scheme becomes preferable, and the entire notion of in-network data aggregation ceases to be useful, in hostile environments.

1.1.1 Related Work

Chan et al. [CPS06] propose a fully distributed aggregation verification algorithm, called the Secure Hierarchical In-network Aggregation (SHIA), which detects the existence of any misbehavior in the aggregation process. The algorithm perfectly satisfies its objective as a detection mechanism; however it is not intended to address our problem as it aims neither at the identification and removal of adversary nodes nor at providing continuous, but gracefully degraded, service under attack. Similarly, the work of Frikken and Dougherty [FD08], which improves the performance of SHIA, aims only at the detection of attacks against the aggregation process.

In contrast to SHIA, Hu and Evans [HE03] and Yang et al. [YWZC06] propose detection algorithms that also allow identification of corrupted nodes. However because both approaches use centralized verification, the incurred communication cost approaches that of the baseline scheme—$\mathcal{O}(n)$ for a network of size n—when in-network data aggregation ceases to be useful. In contrast, the cost of our scheme is logarithmic in n.

Another solution which uses a centralized approach is proposed by Haghani et al. [HPP$^+$07] who extend SHIA. A corrupted node is detected via successive polling of the layers of a commitment tree (generated during the aggregation process) by the BS. Although this work is closest to ours in spirit, it differs in three fundamental ways. First, it incurs a high cost as it not only relies on centralized identification but also each run of the algorithm identifies only one malicious node at a time. In the worst case, to detect c malicious nodes in a network of size n, $\mathcal{O}(nc)$ messages are generated per link. Second, the performance analysis and adversary model presented [HPP+07] does not include a comparison with the baseline scheme where identification of adversary nodes incurs a cost of only $\mathcal{O}(n)$. Hence, it is unclear at what point the proposed scheme ceases to be useful and the baseline scheme becomes preferable. Finally, Haghani *et al.* do not provide network service during the period of the attack.

Group Testing. The identification of corrupted nodes is directly related to the problem of group testing, which strives to identify defective items of a given set through a sequence of tests. Each test is performed on a subset of all items and indicates whether the subset contains a defective item. In combinatorial group testing, it is assumed that the number of defectives in a set is constant. This number can be either known or unknown at the time of testing. Group testing is efficient when the number of defectives in a sample space is small compared to the total number of samples [DH00]. This is an analogous setup to untrusted sensor networks which are characterized as large, densely packed network of sensor nodes.

1.1.2 Our Contributions

In Chapter 3, we propose a *divide-and-conquer* approach to tracing and removing malicious nodes from the network which achieves the three objectives stated above. Briefly, our approach recursively (*i*) partitions suspicious subsets of the network, (*ii*) runs a given 'test' in each partition to check the correctness of the sub-aggregation values, (*iii*) if the result reveals possible node corruption, the set is tagged as suspicious; otherwise, it is considered to be good and the associated sub-aggregate value is retained. Hence, our algorithm allows for the incremental reconstruction of lost data from sub-aggregated value, over the course of its execution. The algorithm terminates when it has isolated all the malicious nodes in the network. The partition test is a primitive which we use in secure aggregation.

The identification algorithm is designed and optimized with respect to the communication cost for an arbitrary number of malicious nodes. We prove the correctness of the algorithm and evaluate its performance using an analysis method inspired by the field of combinatorial group testing [DH00]. Our results illustrate the relationship between the efficiency of malicious-node identification and the number and distribution of these nodes. In particular, we define a precise cost-based threshold when in-network data aggregation ceases to be useful in hostile environments.

1.2 Privacy in Secure Data Aggregation

The principal approach in verifying the integrity of data aggregation is to recompute the aggregate *using the original, raw sensed (measured) data*, and verify that the alleged aggregate is identical or close enough to the recomputed value [DDHV03, HE03, PSP03, GSW04, YWZC06].

This type of verification however is in *direct conflict* with data privacy protection. By allowing a verifier access to the original data, all privacy is lost. We

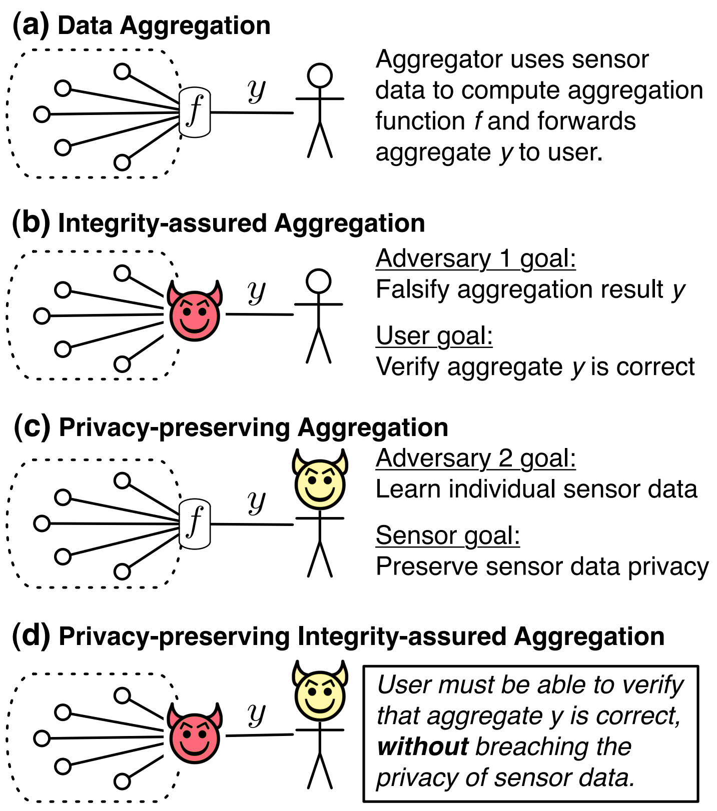

(a) Data Aggregation
f
y
Aggregator uses sensor data to compute aggregation function f and forwards aggregate y to user.
(b) Integrity-assured Aggregation
y
Adversary 1 goal:
Falsify aggregation result y
User goal:
Verify aggregate y is correct
(c) Privacy-preserving Aggregation
f
y
Adversary 2 goal:
Learn individual sensor data
Sensor goal:
Preserve sensor data privacy
(d) Privacy-preserving Integrity-assured Aggregation
y
User must be able to verify that aggregate y is correct, without breaching the privacy of sensor data.

Figure 1.2: The problems of (b) data aggregation integrity and (c) privacy-preserving aggregation have previously been considered as independent problems. In this work, we analyze the relationship and tension that exists when (d) these two problems interact.

consider a scenario where a WSN is deployed remotely from a querying user (Figure 1.2a). The owner of the network controls the access of the user to the network data by allowing the user to learn only specific statistics of the data. Here data aggregation is also used as a *privacy* primitive since it controls the disclosure of the information measured in the network. The user wants to verify the integrity of the received aggregate y (Figure 1.2b) and to perform such verification, the user needs access to individual sensor data. At the same time, however, the network owner does not want the user to access the raw data for privacy reasons (Figure 1.2c). We are interested in the tension that results between preserving data privacy and verifying aggregate integrity (Figure 1.2d). We refer to this problem as the *Privacy-preserving, Integrity-assured data Aggregation (PIA) problem.*

Privacy. Preserving privacy is important in applications where the behavior of individuals or businesses can be deduced from sensed data in a monitored environment. It is often easy to deduce sensitive information based on seemingly innocuous sensed data; e.g., the behavior of a household can be tracked or profiled by monitoring their electrical usage [Har89]. Since sensing is in effect a passive act—as it can be done without the knowledge of the observed party or environment—the observed party has very limited ability to control the extent of information that is disclosed by the sensors. Hence maintaining the privacy of sensed data can be important to parties in the sensed environment. Similarly, sensor data privacy can be important to a service provider that provides service based on the status of the network-sensed data. The service provider may want to guarantee the privacy of the observed party as it can face liability from regulators and customers. Hence both service providers and their customers must rely on the network owner/operator to provide sensor-data privacy. Unless privacy challenges are addressed by the sensor network owner/operator, wide-spread adoption of WSNs in many applications will be impeded by both the customers' and the service-providers' legitimate concerns.

1.2.1 Motivating Example

To further motivate the PIA problem, consider the advanced metering infrastructure (AMI) [Fol08]. An AMI is a system that consists of a large scale sensor network of advanced meters that measure a consumer's energy usage. This information is then sent via wired or wireless links to utilities and service providers, and used for example (*i*) to assist in a change in energy usage of the customers from their normal consumption patterns as a response to changes in price or (*ii*) to service the customer with a more efficient and reliable energy service. Because of the large amount of data that is generated as well as the fact that in many scenarios service providers only require high-level statistical information, the measured data is aggregated before being transmitted to the providers. Since decisions that providers make based on collected information have great economic promise, it is crucial for providers to ensure that the measured data is aggregated correctly.

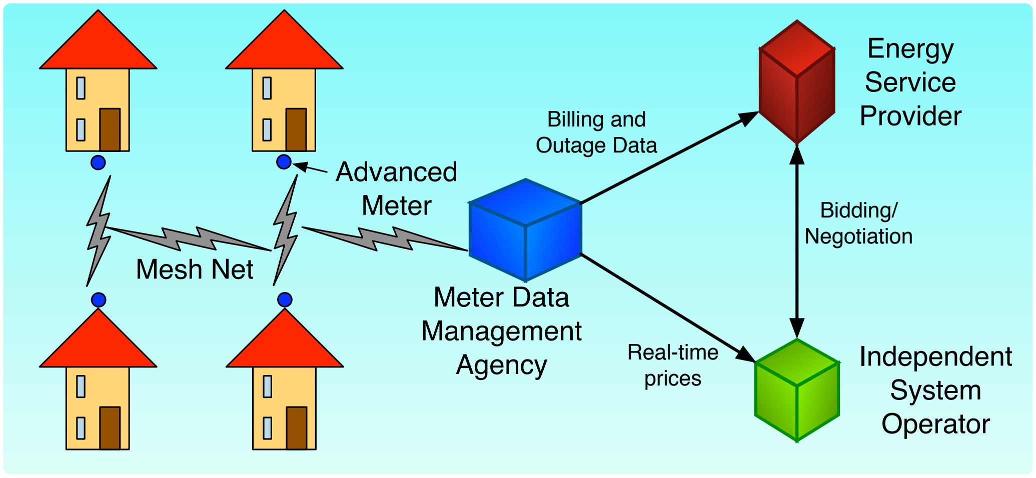


Figure 1.3: An example AMI architecture

At the same time, utilities and service providers in AMI systems must handle privacy concerns from customers and political opposition [Fol08]. AMI directly puts privacy interests at risk because its core purpose is to collect information related to a particular household or business. Meters can collect usage data as well as detailed status and diagnostic information from networked sensors and smart appliances. These data can show directly when and where people are present, as well as what they are doing. Hence, providers must be able to verify the integrity of data aggregation without breaching user privacy.

In this paper, we explore the inherent tension that results between preserving data privacy and verifying aggregate integrity. That is, we aim at providing solutions that enable both aggregate-integrity verification and preservation of sensor data privacy.

1.2.2 Related Work

Existing security solutions in data aggregation deal with two general problems: confidentiality and aggregation integrity. The objective of data confidentiality in aggregation is to ensure that the data transmitted by sensor nodes cannot be discovered by unauthorized entities in the system.

Various solutions have been proposed for preserving aggregate-data confidentiality, including handling hop-by-hop [WGS06] and end-to-end confidentiality [CMT05]. These solutions, however, neither preserve the integrity of the aggregated data nor provide mechanisms for detecting aggregate-data corruption. That

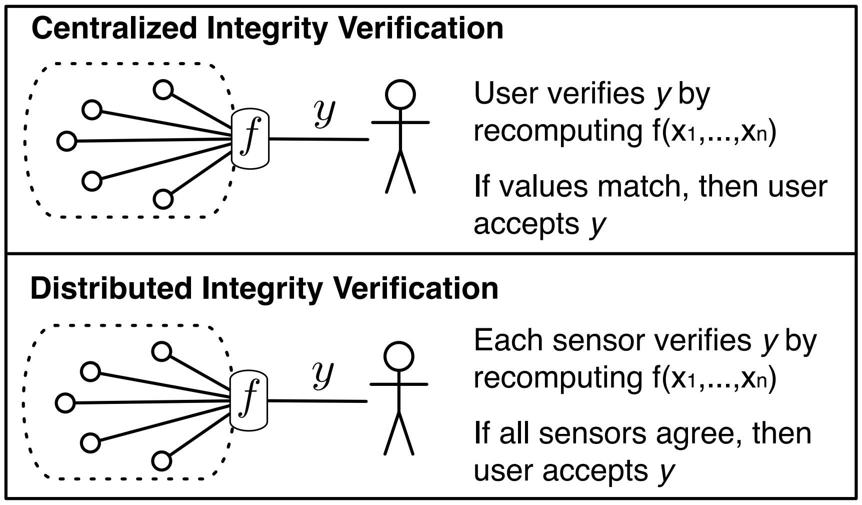

Centralized Integrity Verification
f
y
User verifies y by recomputing f(x_1,...,x_n)
If values match, then user accepts y
Distributed Integrity Verification
f
y
Each sensor verifies y by recomputing f(x_1,...,x_n)
If all sensors agree, then user accepts y

Figure 1.4: The difference between centralized and distributed aggregation integrity verification models.

is, a user that receives an aggregate is unable to verify the validity of the aggregation process and ensure that a malicious aggregator node has not compromised the process.

Various solutions for aggregate-integrity verification have been proposed in both the single aggregator model [PSP03, DDHV03, ZDL06, Wag04] and the "in-network" aggregator model [HE03, CPS06, GSW04, YWZC06, DDHV03, TG08], which uses a hierarchy of aggregators. These solutions fall into two classes: centralized and distributed integrity verification (see Figure 1.4). Although the integrity of the aggregate is verified in both classes by recomputing the aggregation function using the raw data, the classes differ in *who* performs the aggregation recomputation.

In **centralized verification** [HE03, PSP03, GSW04, YWZC06], the querying user receives the raw data to determine if the aggregate has been computed correctly. In contrast, in **distributed verification** [CPS06, FD08], the sensor nodes

	Privacy	Integrity Verif. Cost	Functions Supported	Scheme
Centralized Integrity Verif.	perfect	O(1)	homomorphism based	1
	distribution		comparison based	2
Distributed Integrity Verif.	perfect	O(n)	all	3.1
		O(log n)	decomposable	3.2

Figure 1.5: Summary of PIA solutions, where n is the network size. 'Integrity verification cost' upper bounds the number of messages received and transmitted by each sensor.

themselves recompute the aggregation function using the measurements of the other sensors. If all nodes agree with the same result, the aggregation is considered secure. The main advantage of the distributed approach to centralized is the spreading of the communication through out the network for in-network aggregation.

1.2.3 Contributions

In Chapter 4, we define the problem of Privacy-preserving Integrity-assured Aggregation (PIA). Although the problems of aggregation integrity verification and privacy preservation have been independently considered in the past, this is the first work that analyzes the interaction of these two problems. We analyze the attack models and derive the security requirements, focusing on the single aggregator model (Figure 1.2a).

We propose four distinct symmetric-key solutions that address this problem.

Our results, summarized in Figure 1.5, clearly show that there is an intrinsic tension between providing sensor data privacy and integrity verification algorithms.

In the centralized integrity verification model, the tradeoff appears between the measure of data privacy and the set of aggregation functions that the integrity assurance algorithm supports. In particular, to achieve perfect data privacy—i.e., not leak *any* data—we cannot use a large subset of aggregation functions, including comparison based functions such as maximum. We can only support comparison based functions if we weaken the privacy requirement and preserve privacy of only the distribution of the data.

In the distributed verification model, privacy is traded off for an expanded range of aggregation functions supported by the integrity-verification algorithm, as well as its associated communication cost. For example, to support all possible aggregation functions and maintain privacy, the cost of aggregation becomes so high that the assumed aggregation benefits (i.e., reduced communication cost) disappear.

1.3 PIA Problem in Publish-Subscribe Systems

The privacy-preserving integrity-assured data aggregation (PIA) problem is relevant in many publish-subscribe (pub-sub) system. A pub-sub system [BHGB07, CRW01, RPS06, SBC+98] consists of a large number of publishers submitting information to the system and a (smaller) set of subscribers who register to receive publications of interest. Such systems often cover wide-area networks and involve thousands of publisher as well as routing nodes that forward published data to tar-

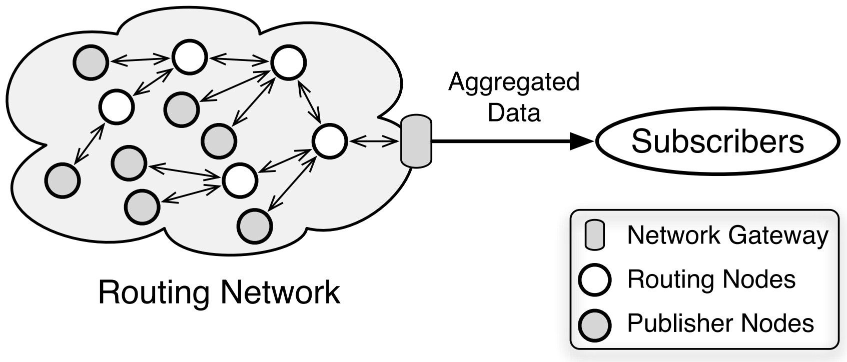


Figure 1.6: Publisher-subscriber model

get subscribers. Pub-sub systems have many applications, from smart electric grid systems [sma] to stock quotes to building management systems (BMS) [KNSN05].

Data aggregation is important in such systems because it can improve both the privacy and the efficiency of the system. In many scenarios, subscribers need only high-level statistical information derived from the publishers' raw data. For example, BMSs keep track of aggregate occupancy information for sections of buildings. Furthermore, it is possible that publishers do not want to disclose their confidential raw data to the subscribers. Computing and disseminating only the aggregate of the data—that is, removing all but the necessary information—allows the privacy of the published data to be observed against unauthorized subscribers. The efficiency benefits of data aggregation in the routing network of the pub-sub system is also important as pub-sub systems can consist of thousands of publisher nodes.

The PIA problem is a major challenge in such aggregation systems. Since wide-area pub-sub systems usually span multiple administrative domains, publisher, routing and subscriber nodes do not always trust each other. A malicious (routing) aggregator node can effectively control the data of all the publisher nodes in its aggregation subtree. Such an attacker can then trick the monitoring applications (that act as subscribers) into making unsafe control decisions by providing incorrect aggregates. Subscribers therefore need to ensure that the data aggregation process performed by the untrusted routing nodes is correct; i.e. ensure the integrity of the aggregation process. This verification needs to be done such that the confidentiality of the raw data from the publishers is preserved. In other words, subscriber should be able to verify the integrity of the aggregate without the publishers' raw data.

In this part of the thesis, we study the PIA problem for *multiple subscribers* in a pub-sub system. In particular, we consider a setting where multiple subscribers need to verify the correctness of an alleged data aggregate in a competitive environment—that is, a subscriber who gets this information before others can gain a decided competitive (e.g. economic) advantage.

As an example, consider the pub-sub system used in the smart electric grid system. Here publishers are electricity meters in many home (where the meters are connected to a wide-area network), and each publisher (i.e., a meter) reports the real-time power demand of each household. Subscribers of the pub-sub system are electricity suppliers that receives the aggregate power demand of each geographical region in a wide service area. Since those electricity suppliers compete with each other in the electricity market, a supplier of electricity can get a distinct economic advantage if it obtains a true aggregate power demand in a timely manner when others do not. For example, such a supplier can change the price structure for

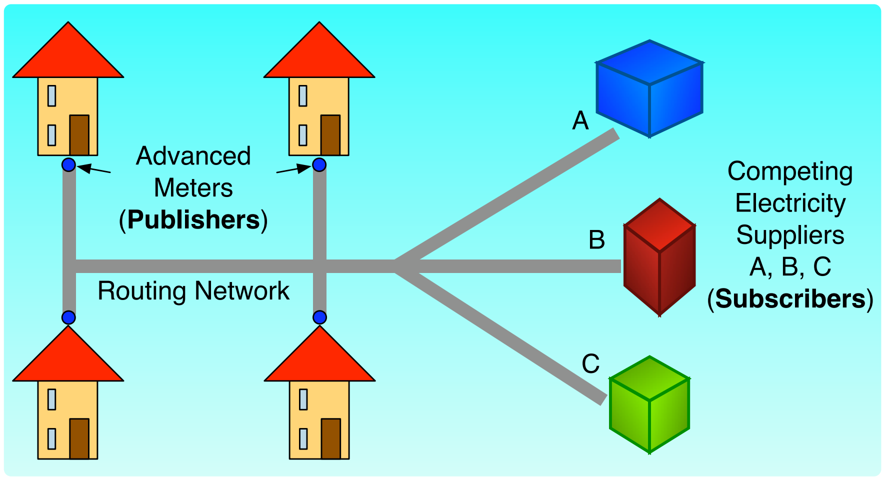


Figure 1.7: A smart electric grid with multiple competing electric suppliers.

electricity by considering a rapid shift of the power demand.

When we provide an integrity-assured aggregate to multiple subscribers, we need to consider a new threat model with malicious subscribers. A malicious subscriber can perform a wide variety of attacks. For example, it can try to cheat the other subscribers by convincing them that a valid aggregate is actually invalid. It can also mount a denial-of-service attack by stopping the validity proof to go through. This attack is specially strong when the adversary is stealthy, i.e., can escape detection. In this problem, we focus on a *stealthy attacker whose goal is to cheat by convincing other subscribers in the system of a false aggregation integrity verification result and do this without detection.*

1.3.1 Prior Work

Although many researchers [Khu05, Mik02, PEB06, PEB07, OP01, RR06, SL07, WCEW02, ZS06] have studied security issues for pub-sub systems, there is very little security research for those that support in-network aggregation. Ahmad et al. [AK06] developed a secure additive aggregation protocol in a large-scale overlay network. Their protocol uses an additively homomorphic public-key cryptosystem to protect confidential data from intermediate aggregation nodes. However, their scheme does not address the issue of integrity discussed in this paper.

No existing aggregation solution considers either collaborative subscribers or fairness in aggregation verification. The work closest to ours in spirit is by Minami et al. [MLWB08] who recently proposed an integrity-assured data aggregation scheme for pub-sub systems. Each subscriber can verify the integrity of the aggregation process independently. Since the proposed system also requires all subscribers to share the same secret, a malicious subscriber can convince other subscribers of an invalid aggregate value. The authors address this problem using a delayed verification approach in which a trusted third party (TTP) synchronizes the disclosure of a secrets for verifying an aggregate to each subscriber at each round of publication. The existence of such a TTP and the synchronous communication channel may not be practical in scenarios consisting of multiple untrusted domains. Furthermore the TTP introduces a single point of failure in the system, which we want to avoid.

In the area of sensor networks, a number of secure aggregation schemes [CMT05, CPS06, GSW04, HLN^{+}07, PSP03] have been proposed. None of the solutions deal with the problem of multiple sink nodes (i.e., base stations). Furthermore, a sensor network has a different communication model from that of a pub-sub system. In particular, all entities, including sensors, base station and the querying user have bidirectional communication channels and the base station or the user can use an authenticated broadcast. In contrast, an authenticated broadcast is inefficient in a pub-sub system, and it could cause a long latency for the delivery of an aggregate. Also, many pub-sub systems define privileges for the publication of data to prevent an adversary from performing a denial-of-service attack by sending bogus messages. Such pub-sub systems assume unidirectional communication between the routing network and the subscribers, that is, subscribers cannot send messages to publishers or routing nodes of the pub-sub system.

An interesting approach towards integrity-assured data aggregation in sensor networks is the distributed verification approach proposed by Chan et al. [CPS06]. Chan et al.'s scheme, SHIA, allows a BS to verify the integrity of the in-network aggregation process within a sensor network by sharing the responsibility of the verification by all the sensor nodes in the network. SHIA, however, requires communication from a BS to the sensor network, while a pub-sub system does not usually support such backward communication channels from subscribers to routing nodes of the system because of various reasons such as efficiency. Additionally, SHIA does not consider the case where multiple base stations (i.e., subscribers) receive the same aggregate data.

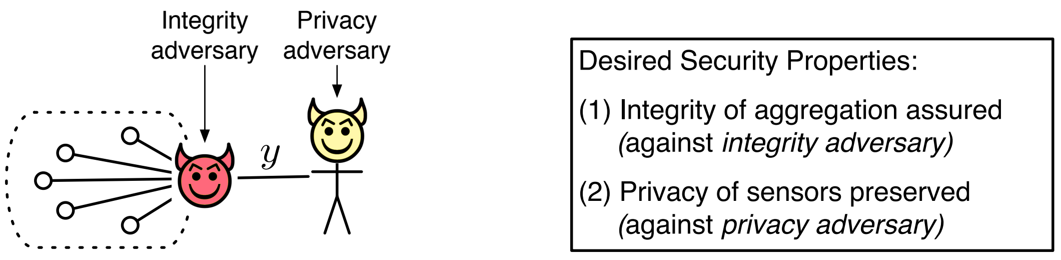
Haber et al. [HHSY06] address the problem of verifying the integrity of aggregate queries on outsourced databases. They develop a verification protocol that allows a user to verify the integrity of the sum of multiple values in a database without seeing those individual values. This query is processed by an untrusted service provider that is different from the trusted database owner. Since a single database owner provides all of the individual values for the sum, their protocol can ensure the integrity of the sum by providing the user with a Merkle hash tree of commitments to the individual values, so that the user can verify the authenticity of those commitments with a digital signature on the root node of the hash tree created by the database owner. This solution of constructing a single digital signature on multiple commitments is not applicable to our problem, since each individual data is provided by a different publisher.

1.3.2 Our Contributions

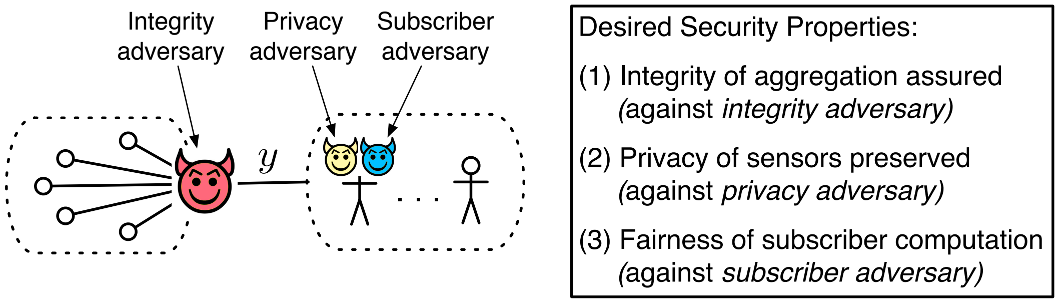
Our goal in Chapter 5 of this thesis, is to design a privacy-preserving integrity-assured aggregation solution that is both collaborative and fair. A scheme is *collaborative* if a subscriber can verify the validity of an alleged aggregate if and only if when all other subscribers are available and have received the alleged aggregate and proof of aggregation correctness. We introduce the notion of *fairness*: subscribers verify the integrity of an alleged aggregate so that they can either all correctly verify the aggregation process or they can detect all cheating subscribers.

We develop a new PIA protocol for multiple subscribers in a pub-sub system. Our scheme is based on the aggregate-commit-prove framework of Chan et al. [CPS06] and Przydatek et al. [PSP03] where each node constructs an aggregate value and commitment structure that incorporates their own published data. Our scheme allows multiple subscribers to verify the integrity of an aggregate in a collaborative and fair way, without assuming to have a trusted third party. Our scheme focuses on the SUM aggregation function, which extends easily to other aggregation functions, such as COUNT and MIN/MAX [CPS06].

In our aggregation protocol, secret key shares are distributed amongst the subscribers so that when the shares are combined together, the integrity of the aggregation process can be verified. This is equivalent to an (n, n)-threshold secret sharing scheme. Thus, a group of subscribers can verify the integrity of an aggregate if and only if every subscriber collaborates following our protocol; a strictly smaller subgroup cannot learn any information about the integrity of the aggregate. Furthermore, publisher and routing nodes construct a commitment value for each subscriber that forces them to act honestly. If any subscriber misbehaves and does not follow the protocol, they are *detected and identified* by the other subscribers.


Integrity adversary
Privacy adversary
y
Desired Security Properties:
(1) Integrity of aggregation assured (against integrity adversary)
(2) Privacy of sensors preserved (against privacy adversary)

(a) PIA adversary and security models


Integrity adversary
Privacy adversary
Subscriber adversary
y
Desired Security Properties:
(1) Integrity of aggregation assured (against integrity adversary)
(2) Privacy of sensors preserved (against privacy adversary)
(3) Fairness of subscriber computation (against subscriber adversary)

(b) PIA adversary and security models for multiple users

Figure 1.8: PIA adversary and security models for (a) single and (b) multiple subscriber model.

We propose a new security model with three types of adversaries: (*i*) a stealthy integrity-adversary that compromises aggregating nodes with the aim of falsifying the aggregation result; (*ii*) a privacy-adversary that compromises subscribers with the aim of learning the raw published values; and (*iii*) a stealthy subscriber-adversary that compromises subscribers with the the aim of falsifying the aggregation verification result. This model is summarized in Figure 1.8. We analyze our protocol in this security model and formally prove the security of our scheme in this model.

1.4 Thesis Organization

This thesis is organized as follows. In Chapter 2.1, we present a short overview of wireless sensor networks and the role and modeling of data aggregation in sensor networks. In Chapter 3, we address the problem of resilient data aggregation in an adversarial setting. Chapter 4 addresses the problem of privacy-preserving integrity-assured data aggregation (PIA) and in Chapter 5 we look at the application of the PIA problem in a publish-subscribe system.

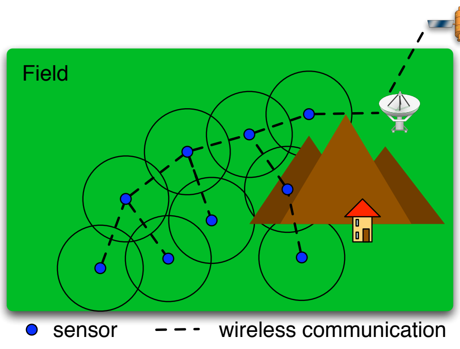
Chapter 2

Preliminaries

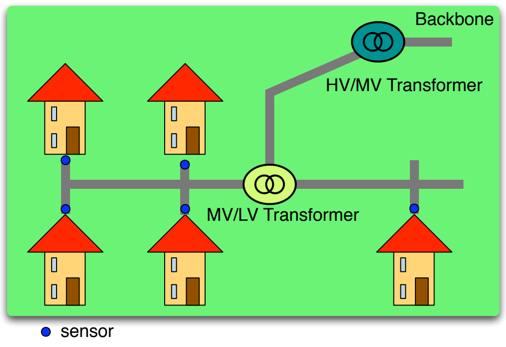
2.1 Wireless Sensor Networks

Wireless sensor networks (WSN) are ad hoc networks formed by typically large numbers of small sensing devices with the ability to communicate amongst themselves and also with an external base station. The sensors could be scattered randomly in harsh environments such as a battlefield or deterministically placed at specified locations. The sensors coordinate among themselves to form a communication network such as a single multihop network or a hierarchical organization with several clusters and cluster heads.

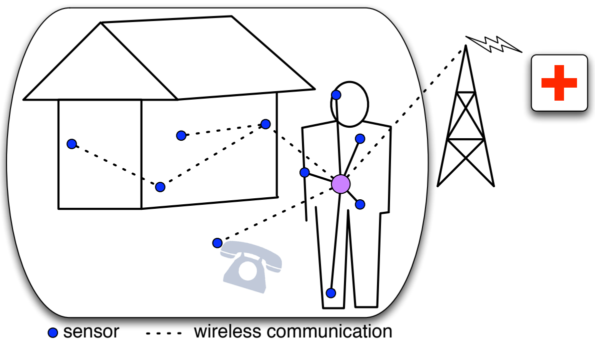
The development of WSNs was originally motivated by military applications such as battlefield surveillance. However, wireless sensor networks are now used in many civilian application areas where there exists distributed interaction between sensors and the physical environment. Example applications include (*i*) wildfire tracking and monitoring [MCP+02, DS05], (*ii*) surveillance systems and burglar alarms, (*iii*) health and biomedical monitoring systems such as monitoring the health status of cattle stocks on farms [BKM04, SCD+05], (*iv*) disaster recovery systems such as monitoring the health of buildings or incremental shifts of ice and snow in alpine mountains, (*v*) industrial automation or supervisory control and data acquisition systems (SCADA) [KAB+05], (*vi*) traffic control [CCT05] or (*vii*)



(a) Habitat monitoring



(b) SCADA systems such as power grid



(c) Healthcare

Figure 2.1: Applications of wireless sensor networks

military warfare[AGHS01] (see Figure 2.1). Many of these systems consist of a distributed network of nodes that measure (sense) the environment from different vantage points; a global view of the network can be constructed by a base station for an outside user by collecting the sensed data and upon analysis of the emergent properties, appropriate action can be initiated.

In addition to one or more sensors, each node in a WSN is typically equipped with a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. The size of a single sensor node can vary from shoebox-sized nodes down to devices the size of grain of dust. The cost of sensor nodes is similarly variable, ranging from hundreds of dollars to a few cents, depending on the size of the sensor network and the complexity required of individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth. [RM04] provides an overview of the range of current sensor devices. Sensor nodes are generally characterized by their constrained resources, high failure rates and low physical security and unattended operation.

2.2 Data Aggregation Models

WSNs are generally assumed to be large scale networks where large amounts of data can be generated. Since sensor nodes are energy constrained it is inefficient for all the sensors to transmit their raw data directly to the base station. In addition, data generated from neighboring sensors is often redundant and highly correlated. *Data aggregation* attempts to collect the most critical data from the sensors and make it available to the base station in an energy efficient manner. In such a setting, certain nodes in the network, called *aggregators*, collect raw data from the sensors, aggregate the data locally and forward only the data aggregate to the base station.

Network aggregation is generally classed into two distinct models: the single aggregator model and the in-network aggregator model, see Figure 2.2.

The *single aggregator model* considers a setup of n sensor nodes $\mathcal{N} = \{s_1, ..., s_n\}$ and one aggregator node $\mathcal{A}$ which is in direct contact with each sensor. At some

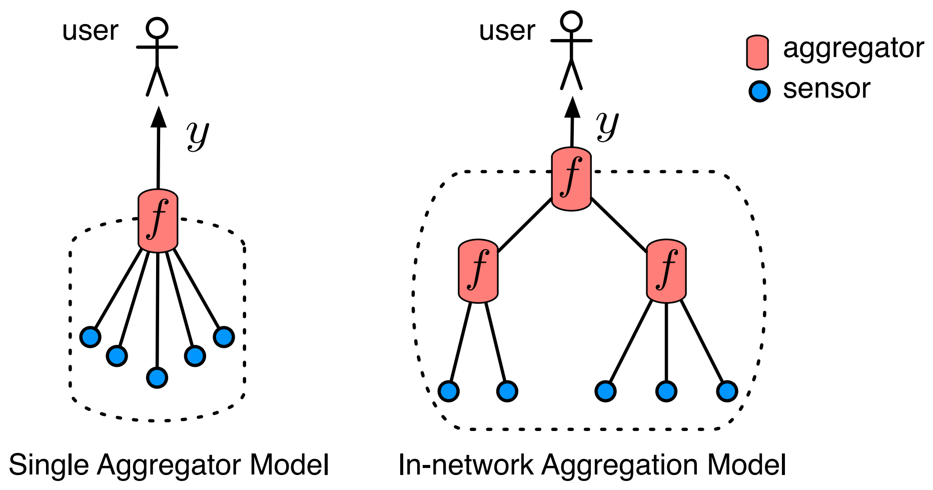


Figure 2.2: Single aggregator and in-network aggregation models

point each sensor node takes a measurement and sends this data to the aggregator. The goal of the aggregator is to compute an aggregate value y that summarizes the sensor readings $x_1, \ldots, x_n$ using the aggregation function f: thus $y = f(x_1, \cdots, x_n)$. The aggregate value is then forwarded to an external user $\mathcal{U}$ or a base station (BS), possibly in response to a query.

This simplified and abstract model of the aggregation network inherently ignores the structure of the multi-hop network and assumes that each sensor node has a separate and authenticated link to either the base station or the user. The resulting communication reduction is limited to the link between the aggregator and the external user. The single aggregator model is therefore not scalable to large sensor deployments.

In order to reduce both energy and communication bandwidth in the network, it is useful to move the integration and filtering of data into the network itself. *In-network aggregation* is a mechanism for adopting multiple and hierarchical aggregators within the network. The *in-network aggregator model* assumes the following.

Consider a multi-hop network of n sensor nodes $\mathcal{N} = \{s_1, ..., s_n\}$ which consists of a set of aggregating nodes $\mathcal{A}$ and a set of sensing nodes $\mathcal{S}$, where $\mathcal{A}, \mathcal{S} \in \mathcal{N}$ and $\mathcal{A} \cup \mathcal{S} = \mathcal{N}$. For simplicity, we ignore other states of nodes within the network such as idle or simply forwarding. We assume that $\mathcal{A} \cap \mathcal{S} \neq \emptyset$ as a sensor can both sense and aggregate. Although the network can contain multiple base stations, we assume that the aggregation round is in response to query from one base station. We present the network using the following parameters: $(\mathcal{N}, \mathcal{A}, \mathcal{S})$.

Aggregation is implemented over a hierarchical routing topology, which can be based on clusters, chains, trees or grids. In general, in-networking aggregation assumes a tree-based topology based on a spanning tree over all the nodes in the network rooted at the base station. The properties of the spanning tree, such as minimum energy or delay, are considered outside the scope of this work.

Chapter 3

Efficient Handling of Integrity Attacks

In this Chapter, we address the problem of integrity attack detection and response in a data aggregation application in WSN.

3.1 System Model

Consider a multihop network of untrusted sensor nodes and a single trusted BS. The system administrator or user that resides outside the network interacts with the network through the BS interface. For brevity, subsequently we refer to any requests made by this external entity via the BS, as simply requests by the BS. We assume that each sensor has a unique identifier v and a unique secret key shared with the BS, K_v. The sensor network continuously monitors its environment and measures some environmental data. We divide time into epochs; during each time epoch, the BS broadcasts a data request to the nodes in the network and nodes forward their data response back to the BS. Data can be forwarded individually or as an aggregate.

We model node corruption in the network as a function of the number c and the distribution of the corrupted nodes. Each sensor node v belongs either to the good set G or the malicious or corrupted set M. A network instance is defined as $N = \{\forall v \text{ in network} : v \in G \vee v \in M\}$ where $|M| = c$ and $G = N \setminus M$. The collection of all N for a given c, constitutes a family of networks $\mathcal{N}_c$.

For the purpose of computing the aggregate, we assume that the sensed environment (e.g., temperature) changes minimally with respect to the duration of the identification algorithm. This is a practical assumption as once malicious activity is detected, the identification algorithm is promptly executed. Moreover the algorithm terminates after a small, constant number of rounds.

3.2 Adversary Model

We assume that the network is deployed in an adversarial environment where the adversary can corrupt an arbitrary number of nodes. Once a node is corrupted, the adversary has total control over the secret data of the node as well as the subsequent behavior of the sensor node. We assume that a corrupted node *persistently* misbehaves by inducing the BS to accept an 'illegal' value. An illegal value is defined based on the adversary objectives which is to induce the BS to accept a data value which is not already achievable by direct data injection.

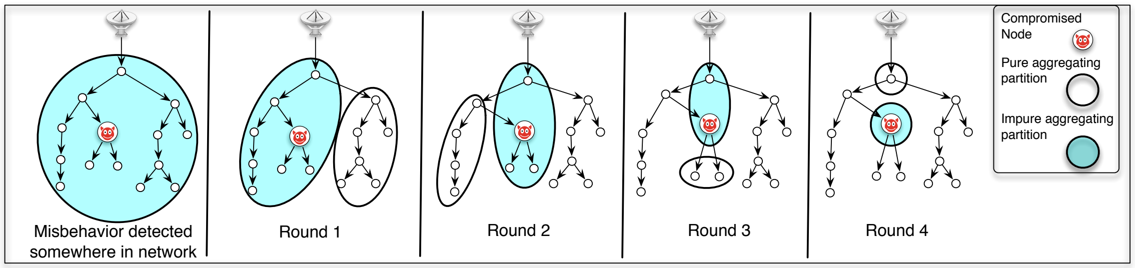
A direct data injection attack occurs when an adversary modifies the data readings reported by the nodes under its direct control, under the constraint that only legal readings in $[r_1, r_2]$ are reported [CPS06]. In the case of a single data values, this means that the data value transmitted is outside the legal reading of $[r_1, r_2]$. This is called a *false data injection attack*. In the case of data aggregation, the objective of the adversary is to tamper with the aggregation process such that the BS accepts an aggregation result which is not achievable by the direct data injection. We refer to this type of attack as a *false aggregation attack*. An aggregation protocol is considered secure if the adversary cannot successfully launch such an attack [CPS06].

3.3 Performance Measure.

We use link cost as a metric to analyze our algorithm. Link cost is defined as the total number of messages transmitted over a particular link in the network and is important as it determines how quickly nodes in the network exhaust their energy supply. Such nodes are often core to the connectivity or the functionality of the network and their loss can lead to network partitioning or denial of service.

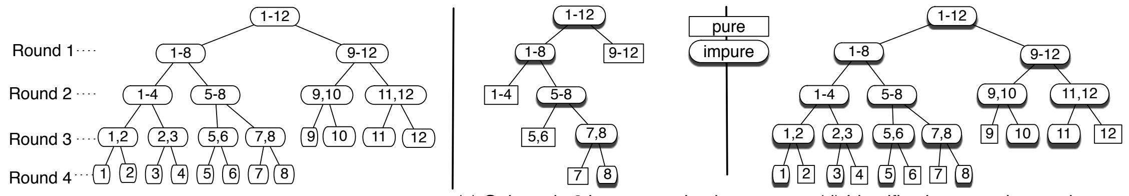
3.4 Identification Algorithm

The main objective of our algorithm is to recursively isolate the malicious nodes in the network and thus render the adversary inoperative. The algorithm is initiated once misbehavior is detected in the network (e.g., via [CPS06]) and is executed over a number of rounds, following an intuitive divide-and-conquer approach. In each round the algorithm partitions the suspicious subsets of the network and performs a partition test on the newly formed groups. The number of subsets a suspicious group is partitioned into is called the *partition degree*. The partition test consists of nodes aggregating their data and verifying the integrity of their aggregation process. The test has two outputs: ‘pure’ if all the nodes in the partition are good and ‘impure’ if there is at least one malicious node in the group. The algorithm terminates when there are no remaining impure groups.



Compromised Node
Pure aggregating partition
Impure aggregating partition
Misbehavior detected somewhere in network
Round 1
Round 2
Round 3
Round 4

(a) Recursive isolation of captured nodes.



(b) The unpruned identification tree

(c) Only node 8 is compromised. Identification corresponds to Fig (a)

(d) Identification tree when nodes 1, 3, 5, 8, 10 and 11 are compromised.

Figure 3.1: Identification algorithm on an input of 12 nodes, $m = 2$.

By distributing the localization of the malicious nodes, the scheme simply keeps track of the lower bound on the number of malicious nodes in the network and increases the bound only when the findings of the scheme up to that point imply that this is valid.

Algorithm 1 *Identification*

Input: *All the nodes in the network $N \in \mathcal{N}_c$, partition degree $m > 1$, where integer m is the number of partitions a group divides into in each iteration.*

Output: *A result set M of malicious nodes and a result set G of good nodes, such that $M \cup G = N$*

Let $t = 1$ be the lower bound on the number of malicious nodes in the network and $S = \cup_{i=1}^{t} S_i$ denote the current set of suspicious nodes, $S_1 = N$.

1. For $j = 1, \cdots, t$, BS requests partition S_j to be divided into m disjoint partitions (using partition rule viz. Algorithm 2). The collection of subdivided sets form the current *collection S. Set t to be the cardinality of set S.*

2. For $j = 1, \cdots, t$, if $|S_j| > 1$, the nodes in partition S_j partition themselves into groups of size $\frac{n}{m}$ and execute partition test. BS verifies the purity of each partition.

3. The BS learns the status of each node for the following round (details are provided in the next section). For $j = 1, \cdots, t$, if S_j is pure (i.e., all the nodes are good), then $G = G \cup S_j$; else if S_j is impure (i.e., there is at least one misbehaving node) and a singleton set, then $M = M \cup S_j$ and decrement t. Adjust the indices of the remaining sets, $\{S_j\}$ appropriately, to include only sets that are impure and non-singleton. If $t > 0$, go to step 1 (next round), else quit as all malicious nodes have been traced.

We can model the divide-and-conquer approach of Algorithm 1 as the pruning process of an m-ary tree T where each tree vertex is associated with a partition test. The root of tree T is associated with the input set N and each round i is associated with level $(i+1)$ of the tree. This is because the identification algorithm is initiated when misbehavior is detected in the network and therefore the test at level 1 has been already executed. If a partition X is tested pure, then all the descendants of the associated vertex are pruned; otherwise the set X is re-partitioned. Fig. 3.1(b) presents an unpruned identification tree for a network of 12 nodes and partition degree $m = 2$. Fig. 3.1(c) and (d) show how the tree can be pruned when the network contains one and six corrupted nodes respectively. Fig. 3.1(a) shows how the identification tree corresponds to the recursive isolation of the captured nodes on the physical network.

Next we define a novel partition rule which generalizes the bijective rule of Du and Hwang [DH93] (see Appendix A). This algorithm partitions the network such that the identification tree contains at most one incomplete subtree. Intuitively a complete tree of n nodes executes less or equal number of tests than an incomplete tree of n nodes as the complete tree contains less vertices (where each vertex corresponds to one test).

```
Algorithm 2 Partitioning Rule
Input: Set $X$, maximum number of partitions $m$.
Output: Result sets $\{X_i\}$, such that $\cup X_i = X$.
Let $i = 1$ denote the new subset ($X_i$) to be determined.
1. Choose $X_i$ to contain $m^{\lceil \log_m |X| \rceil - 1}$ nodes.
2. Update set $X = X \setminus X_i$ to exclude the newly formed subset. If less than $m$ subsets are formed and $X$ has more than $m - 1$ nodes, then increment $i$ and go to Step 1.
Else if $X$ is not a singleton set, increment $i$ and add the remaining nodes in $X$ to $X_i$.
Else if $X$ is a singleton set, then $X$ cannot be partitioned anymore.
```

3.4.1 Partition Test

The test that nodes perform in each newly formed partition is a fundamental step in our algorithm. There are two types of tests depending if the partition is a singleton or otherwise.

Tests for Non-singleton Partitions. In all non-singleton partitions (partitions containing more than one node), data is aggregated and the partition leader directly transmits the partition aggregate (via multi-hop) to the BS, which verifies the integrity of the aggregation process and hence the integrity of the nodes within that partition. In the general case, Algorithm 1 can be composed with any aggregation-verification algorithm that does not depend on a fixed partition and provides provable guarantees. Next we show how we can modify SHIA to satisfy these conditions.

SHIA extends the aggregate-commit-prove framework of [PSP03]. In the aggregate-commit phase of the algorithm, a cryptographic commitment tree (hash tree) is built based on the sensor readings and the aggregation process. This forces the adversary to choose a fixed aggregation topology and set of aggregation results. In the prove phase of the algorithm, each sensor independently verifies that the final aggregate has incorporated its sensed reading correctly. Specifically each sensor reconstructs the commitment structure and ensures that the adversary has not modified or discarded the contributions of the node.

SHIA cannot be used as is because it assumes that the BS knows the exact set of nodes which are alive and reachable. Instead, we propose a new algorithm Group SHIA (GSHIA) which includes two additional properties. First, nodes can organize themselves into groups of size g, where g is arbitrarily defined by the BS. This can be easily achieved as the 'delay aggregation' approach of SHIA develops an aggregation tree one node at a time. Since the root node of the aggregation tree knows the size of its subtree, it can declare a partition complete when it has g nodes or it cannot add any more nodes to its partition.

In GSHIA, the BS can also verify the integrity of the aggregation process for a group of unknown size and membership set. This property can be implemented through the use of a Bloom filter [Blo70] that summarizes the membership information of the partition. The BS then verifies the membership set by exhaustively searching through the possible nodes. The change we propose places most of the membership resolution burden on the BS, which is generally assumed to be powerful. However we can reduce the computation burden by noting that Algorithm 1 is *nested* (i.e., each new partition is a proper subset of an older impure partition) and therefore the space of possible partitions in each round is reduced by a factor of m. Further improvements can be made if the BS knows the topology of the network a priori, using efficient schemes such as [SBD02]. For protocol details as well as analysis and further improvement strategies, we refer the reader to [TG08].

An alternative approach to the above modification is to use the original SHIA algorithm and make the additional assumption that the BS knows the topology of the network prior to the detection period. The BS can then deterministically partition the network for a given m and transmit this information to each sensor. When an impure group is detected, nodes divide themselves according to the specified partitioning. Although this method is simpler and more efficient, the additional assumption is not always practical as sensor networks often have dynamic topologies due to the short life span of the sensors.

Tests for Singleton Partitions. If a partition contains exactly one sensor node, the node v transmits its measured data x_v along with a MAC tag σ_v computed using K_v. Upon receiving $\langle v, x_v, \sigma_v \rangle$, the BS verifies the tag and ensures that x_v is valid. The BS assumes node v has misbehaved if x_v is not in the correct range but the tag verifies correctly.

3.4.2 Group SHIA

For the sake of brevity, we only describe the differences between SHIA and GSHIA and we refer the reader to [CPS06] for details and analysis of SHIA. GSHIA has four main phases: query dissemination, grouping, aggregation-commit and result checking.

Query Dissemination. The request message the BS broadcasts also includes the following grouping information: the size of the new partitions as well as the ID of the groups that were found impure in the previous round.

Aggregation-Commit. Nodes within a group compute a cryptographic commitment structure over their data values and the aggregation process as in SHIA. Also to allow the resolution of the group memberships, each leaf vertex also computes a Bloom filter that probabilistically summarizes the node membership set. The filter is then forwarded to the parent internal vertex, who aggregates the filters using the bit-wise OR function. Additionally each leaf vertex computes an authentication tag over a fixed message and forwards this to its parent node. The tags are aggregated using the bit-wise XOR function to form the *group tag*. The group filter and tag are used by the BS to determine the group membership sets.

Grouping. In this phase, node grouping is conducted through the selection of leader nodes for each group. This phase is executed in parallel with the aggregation-commit phase. Whenever a node performs the aggregation and commitment operations, it also determines if it is a group leader by comparing the group size with the target size broadcast by BS in the request message. If a node is selected as group leader, then all the nodes in its subtree which do not as yet belong to a group become its group members. The group leader then computes a message authentication code (MAC) and forwards the group aggregates and commitments along with the MAC to the BS without further aggregation along the way. If the node is not selected a group leader, it simply forwards its aggregation and commitment values to its parent node, where they are aggregated further. In this way, groups are iteratively generated starting from the leaf nodes and approaching the BS or the root of the network spanning tree. At the end of the aggregation-commit process, all remaining nodes which do not belong to a group are grouped together with the BS acting as their group leader.

Result Checking. At the end of the aggregation-commit and grouping phases, each group leader has reported their aggregation results and commitment values to the base station. The base station first determines the group size and the membership set of each group. This is done by narrowing down the potential membership sets of a group based on the location of the group leader, the group size and the group Bloom filter. The correct membership set can be verified by the aggregated group tag. Once the membership set of a group has been determined, the group size can also be verified. The BS then authenticates the final commitment values of each group and disseminated them to the respective groups.

The result checking is the distributed verification process as in SHIA, where each group verification code is forwarded to the BS by the group leader. When the BS receives all the group confirmation codes, it accepts the group aggregates if it can verify that all group members have individually verified the correctness of the aggregation protocol. The BS discards any group aggregate which is not verified by all group members and classifies the associated group as suspicious. The final network aggregate is computed over all groups which are deemed correct.

3.4.2.1 Analysis

Communication Complexity. The aggregation verification process has a link congestion of $\mathcal{O}(\log^2 n)$ where n is the size of the group [CPS06]. GSHIA introduces the following additional messages in the query dissemination and aggregation-commit phases of the scheme: (1) group IDs, (2) Bloom filter output and (3) group tag. Messages (2) and (3) are fixed size and message (1) depends on the number of active partitions in the network, m. Thus the final link congestion for GSHIA is $\mathcal{O}(\log^2 n + m)$.

Security Analysis of Grouping. We must show that a malicious node cannot force the BS accept a false grouping. A group membership set is summarized via the Bloom filter. The BS can verify that the filter output has not been changed by verifying the validity of the group tag, which is the XOR of the MACs of the group members. Assuming that the MAC scheme is secure, then a malicious node can at best forge the group tag with negligible probability. Hence if the BS can verify the group tag associated with the Bloom filter and ensuring that each node in the network belongs to at most one group, it knows with overwhelming probability that the grouping is correct.

3.4.3 Computing Aggregate

An important feature of our algorithm is that the network aggregate can tolerate malicious nodes and in fact, the aggregate degrades gracefully with the attack. In particular, dual to the intuition that the algorithm recursively isolates the corrupted nodes, is that the algorithm also increasingly identifies the uncorrupted nodes in the network. The BS can then use the data from the nodes determined to be uncorrupted to reconstruct the network service.

Recall our assumption that the sensed environment of the network does not change during the protocol execution. Thus we can improve the quality of the network aggregate in each successive round by incorporating the aggregates of newly found pure groups. Algorithm 3 shows how the aggregate is updated when the aggregation function is sum. We can easily extend this to other low-order statistics functions, such as min/max, averaging, etc.

Algorithm 3 *Aggregate Update in Round i*

Input: *Aggregate Ψ_{i-1} from round $i-1$, set $\{\Psi[j]\}$ of the aggregates of all pure partitions from round i.*

Output: *Aggregate Ψ_i of round i, where $\Psi_i = \Psi_{i-1} + \sum_j \Psi[j]$.*

3.4.4 Security Analysis

In the following, we first show the correctness of the proposed algorithm and in Section 3.5, we propose a mathematical framework to analyze the communication cost associated with providing our security solution.

Theorem 3.4.1 *Given an input set of nodes N and partition degree m, Algorithm 1 outputs two resulting sets of corrupt nodes M and of good nodes G, $M \cup G = N$.*

- *(Completeness) If corrupt node $v \in N$, then $v \in M$, i.e. no false negatives.*
- *(Soundness) If node $v \in M$, then v is corrupt, i.e. no false positives.*

Proof: Let T be the identification tree that Algorithm 1 generates. For any corrupted node $v \in N$, any vertex u in T which contains v, tests impure. This is because a corrupted node is persistently malicious and the partition test $t(\cdot)$ is perfect (i.e., the test result is always correct). Each impure vertex in T is either divided into smaller partitions if it is a non-singleton set, or is added to the set M if it is a singleton set. Since the algorithm converges when $t = 0$ or when there are no more impure non-singleton partitions, then by convergence time the algorithm must have found all corrupt nodes and added them to set M. Thus the algorithm is complete.

Additionally, the algorithm is sound since if node $v \in M$, then there exists a vertex u in identification tree T which is associated with a singleton set $\{v\}$ and that $\{v\}$ is impure. Thus v must be malicious. □

Corollary 3.4.1 *Algorithm 1 isolates all c corrupt nodes within $\lceil \log_m |N| \rceil$ rounds.*

We can trivially prove this as the leaf at the highest level of identication tree T denes the round duration of the algorithm. In particular note that T is rooted at a vertex associated with node set N and the root node (at level 1) is processed in round 0. Also in each round, the active vertices are divided into m equal partitions and at most one of the partitions contains a smaller number of nodes. It is thus easy to see that T has leaves on levels $\lceil \log_m |N| \rceil + 1$ and $\lceil \log_m |N| \rceil$.

3.5 A Theoretical Model for Cost Analysis

In this section, we derive the cost associated with the security guarantees of the proposed protocol. We first formulate the communication cost in terms of an optimization problem. We then analytically solve this problem by introducing a novel mathematical framework, inspired by [DH00, FT99] and evaluate our results using an example network of 4096 nodes. For a complete analysis of the problem, finally we look at the best and average case cost of the system.

The link cost of the algorithm is a function of the number of partitions that are generated in each round (referred to as *partition* cost) as well as the aggregation-verification cost of each partition (referred to as the *test* cost of each partition). It is important to distinguish between the two costs because partition cost is characterized solely by the identification algorithm, whereas test cost is a function of the aggregation-verification primitive adopted and can be improved upon. We emphasize that the total cost derived in this section are based on the use of GSHIA as our primitive.

3.5.1 Cost Upper Bound Definition

Let N be a network instance with c corrupted nodes, $N \in \mathcal{N}_c$, input to the algorithm and let the algorithm terminate in $\tau = \lceil \log_m |N| \rceil$ rounds. Let $P(i, m, N)$

denote the number of partitions in round i where each partition is of size $T(j, m, N)$, $j = 1, \cdots, P(i, m, N)$. We formulate the total communication cost $G(m, N, c)$ of the algorithm as:

$$G(m, N, c) = \sum_{i=0}^{\tau} \sum_{j=1}^{P(i,m,N)} T(j, m, N) \tag{3.1}$$

Worst case cost of the algorithm is the maximum cost of the algorithm for all distributions of c corrupt nodes in the network:

$$G(m, c) = \max_{N \in \mathcal{N}_c} C(m, N, c) \tag{3.2}$$

The optimization problem for the identification algorithm is defines as:

$$G(c) = \min_{m>1} G(m, c) \tag{3.3}$$

The parameters which achieve $G(c)$ are called the minimax parameters of the identification algorithm. *The goal of the network administrator is to find the minimax parameter m for a given network N without knowing the number of corrupt nodes c.*

The primary parameter in Equation 3.3 is partition degree m. Towards solving Equation 3.3 we consider the effect of m on the different components of cost. Test cost for singleton and non-singleton groups are $\mathcal{O}(1)$ and $\mathcal{O}(\log g)$ (refer to Appendix A) respectively, where g is the size of the group. Thus test cost is logarithmically related to $1/m$.

In the following, we present some results relating m with partition cost. This is of particular interest as our results can be applied to other divide-and-conquer algorithms. In fact the isolated problem of optimizing partition cost is equivalent to an instance of combinatorial group testing problem, where the number of defectives is unknown and we optimize the algorithm to minimize the number of tests performed. Inspired by group testing results for $m = 2$ [DH93], we extend the results for the general m-ary case. To the best of our knowledge this is the first time the m-ary case has been considered.

3.5.2 Results

3.5.2.1 Upper bound when n is a power of m

We first prove the upper bound of the partition cost for different m-ary identification algorithms, where the number of nodes in the network n is a power of m and then compute the upper bound of the total cost of the identification scheme when n is a power of m.

Theorem 3.5.1 *Let n be a power of $m > 1$. Then for c corrupted nodes in n nodes, $1 \leq c \leq n$, the number of partitions generated is tightly upper bounded by $\frac{m}{1-m} + mc(\log \frac{n}{c} + \frac{1}{m-1})$.*

Proof: Let T be the m-ary identification tree whose root vertex is associated with a set of size n, which is a power of m. According to the algorithm, every internal vertex must be associated with an impure set and there must exist exactly c impure leaves. We sum up the total number of pure leaves in T as follows. Let u denote the height of tree T, $u = \log_m n$. Each level i has m^{i-1} vertices, where at most c are impure. Level $v = \lceil \log c \rceil$ is the first level with at least c vertices and let

$w = v - \log c$. The total number of impure nodes γ in T is:

$$\begin{aligned}
\gamma &= \sum_{i=1}^{v} m^{i-1} + c(u - v + 1) \\
&= \frac{(1-m^v)}{1-m} + c(\log n - (w - \log c) + 1) \\
&= \frac{1}{1-m} + c(\log\frac{n}{c} - w + 1 - \frac{m^w}{1-m}) \\
&\leq \frac{1}{1-m} + c(\log\frac{n}{c} + \frac{m}{m-1})
\end{aligned}$$

since $0 \leq w < 1$ and $f(w) = -w - \frac{m^w}{1-m}$ is a convex function. For $0 \leq w \leq 1$, $f(w)$ is maximized at $w = 0, 1$, where $f(0) = f(1) = \frac{1}{m-1}$. Thus there are at most $\gamma - c = \frac{1}{1-m} + c(\log\frac{n}{c} + \frac{1}{m-1})$ impure internal nodes in T. Each internal node has exactly m children, so T has at most $\frac{m}{1-m} + mc(\log\frac{n}{c} + \frac{1}{m-1})$ nodes. □

Theorem 3.5.2 *Let n be a power of $m > 1$. Then for c corrupted nodes in the n nodes, $1 \leq c \leq n/m$, the total cost $G(m,n,c)$ of the identification algorithm is upper bounded by $\sum_{i=1}^{u} H[i]$ where H is a sequence of length $u = \lceil \log_m n \rceil$:*

$$H[i] = \begin{cases} m^{i-1}(\log\frac{n}{m^{i-1}} + 1) & \text{if} \quad i < v \\ mc\,(\log\frac{n}{m^{i-1}} + 1), & \text{if} \quad i \geq v \end{cases} \tag{3.4}$$

where $v = \lceil \log_m c \rceil$ and log *denotes* $\log_2$.

Proof: Let tree T be the m-ary identification tree whose root vertex is associated with a set of size n. Let sequence element $H[i]$ represent the total cost of the identification algorithm in level i of the identification tree T. Each level i of T has m^{i-1} vertices, where at most c are impure. Also each vertex at level i has exactly $\frac{n}{m^{i-1}}$ nodes. Level v of T is the first level where T has at least c vertices. Therefore at level $i < v$, all m^{i-1} vertices are impure. Since each test has a cost of at most

$(\log p + m)$, where p is the number of nodes tested, the total cost of each level $i < v$ is upper bounded by $m^{i-1}(\log \frac{n}{m^{i-1}} + m)$. Now consider level $i \geq v$. Then each level has at most mc impure nodes of size m^{i-1}. Therefore total cost of each level $i \geq v$ is upper bounded by $mc(\log \frac{n}{m^{i-1}} + m)$. $\square$

3.5.2.2 Upper bound when n is not a power of m

In the general case when n is not a power of m, we cannot use the approach of [DH93] (solved for $m = 2$) as the number of possible ways the corrupted nodes are distributed within each subtree explodes (analogous to the combinatorial, ball in the bucket problem). Instead we propose a novel model, inspired by the work of Fiat and Tassa [FT99] in the context of dynamic traitor tracing (DTT)[1]. We introduce the notion of a *path trace*, defined with respect to a particular corrupted node. The path trace traces the identification path of that node in the identification tree T. Informally we say a path trace D for corrupted node u is rooted at the vertex v in tree T that the identification algorithm separates it from the other corrupted nodes in the network. The trace includes all the vertices in the path between v and the leaf vertex associated with set $\{u\}$. Therefore each time an impure vertex v' in T has more than one impure child, then the algorithm learns that the node set at v' contained more than one corrupted node, and thus a new tree trace D' is generated. Fig. 3.2 shows the paths generated for an example identification tree. Note that although a path trace is not unique to a given node, the set of path traces generated

[1]The DTT model differs to ours as in each of its rounds, only one node misbehaves.

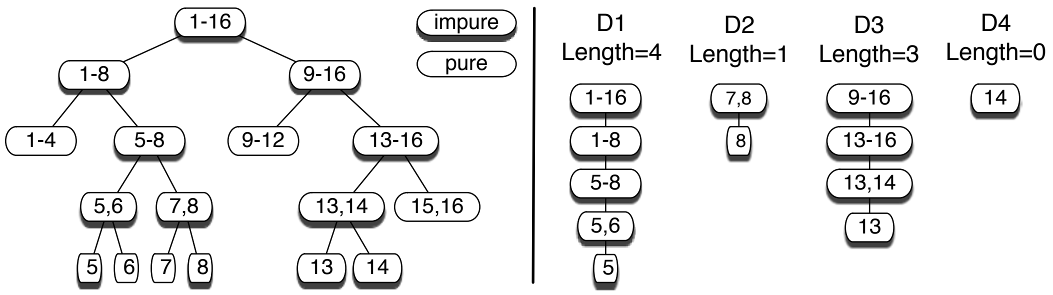


Figure 3.2: Path traces for corrupted nodes 5, 8, 13 and 14.

is unique. We can therefore determine the set of path traces in a network without associating them to a particular corrupted node.

Lemma 3.5.1 *A path trace of length ℓ generates $m\ell$ partitions where there are m partitions of sizes $\{m^{\ell-i}\}$, $i = 1, \cdots, \ell$.*

Proof: The path trace is a path on an m-ary tree and each internal vertex on the path has $(m-1)$ other siblings that are also tested. Also a path trace of length ℓ has a root vertex associated with m^ℓ nodes. Thus at level i of the path, the vertex is associated with $m^{\ell-i}$ nodes. $\square$

Consider identification tree T generated by Algorithm 1, for n nodes.

Lemma 3.5.2 *Let $n = m^h$ where $h \in \mathbb{Z}^+$. Then define sequence P as:*

$$P = \{h, \{h-1\}^{m-1}, \{h-2\}^{m(m-1)}, \{h-3\}^{m^2(m-1)}, \cdots\} \tag{3.5}$$

where $\{y\}^x$ denotes the value y repeated x times. The first c elements in P represent the tight upper bound on the length of the path traces generated when n contains c corrupted nodes.

Proof: Since n contains at least one corrupted node, the first path trace D_1 is rooted at the root of T and thus has length h. A new path trace is generated any time a vertex in T contains more than one impure child. To find the upper bound on the length of the path traces, each trace should be generated in as early a round as possible. On level 2 of T (round 1), up to $(m-1)$ path traces can be generated of length $(h-1)$; at level 3, up to $m(m-1)$ path traces can be generated of length $(h-2)$, and so on. In general, in level i, up to $m^i(m-1)$ path traces of length $(\log_m n - (i-1))$ can be generated. Since one path trace is associated with each corrupted node and there are c corrupted nodes, the set of lengths associated with the generated path traces can be represented by the first c elements of P. $\square$

Theorem 3.5.3 *Let $m^{h-1} < n < m^h$ where $h \in \mathbb{Z}^+$. Then define sequence P' as:*

$$P'[i] = \begin{cases} P[i] & \text{if} \quad i < x \\ P[i]-1 & \text{if} \quad (i > x \,\&\, P[i] > 0) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where P is the sequence defined in Equation 3.5 and the index

$$x = \left\lceil \frac{n - m^{h-1}}{m-1} \right\rceil \tag{3.7}$$

The first c elements in P' represent the tight upper bound on the length of the path traces generated when n contains c corrupted nodes.

Proof: It is trivial to show that the number of internal nodes at level $h-1$ is defined by x. Then there are $(m^{h-1} - x)$ leaves in level $(h-1)$ and $(n - m^{h-1} +$

x) leaves in level h. To find the upper bound of the lengths of the path traces, $\min(x, c)$ of the path traces are associated with a corrupted node at level $(h + 1)$ and thus they correspond to the identification tree for m^h nodes. The path traces corresponding to the remaining corrupted nodes have leaves at level h and correspond to a identification tree for m^{h-1} nodes. $\square$

We can use Theorem 3.5.3 to derive the tight upper bound on the total cost of the identification algorithm. Consider identification tree T generated by algorithm 2, for a network of n nodes. Let T contain α complete m-ary trees and one incomplete m-ary tree, with respective depths $d_1, \cdots, d_{\alpha+1}$. Let $P_1, \cdots, P_{\alpha+1}$ correspond to the set of potential path traces for each of the $(\alpha+1)$ respective subtrees using Lemmas 3.5.1 and 3.5.2. Then let sequence P be composed of the non-increasing ordered set of the path traces $\{P_i, \cdots, P_{\alpha+1}\}$, i.e. $P = \{h, (h-1)^{a_1}, (h-2)^{a_2}, \cdots\}$, where $a_1, a_2, \cdots$ are dependent on the size of the subtrees. If n contains c corrupted nodes, then the length of the generated path traces are bounded by the first c elements in P. We can use Lemma 3.5.1 and sequence P to compute the size and number of the partitions that Algorithm 2 generates in the worst case and derive total cost by summing the cost of the c path traces.

Finally we derive a closed form expression for the *loose* upper bound on the total cost of the network. This is purely for the purposes of comparison of our work with existing solutions.

Theorem 3.5.4 *For c corrupted nodes in a network of size n, the identification algorithm has a communication link cost of $\mathcal{O}(c^2 \log^3 n)$.*

Proof: Consider the detection tree corresponding to the identification algorithm, which is of height $\log n$ (Theorem 3.4.1). Then for the worst distribution of compromised nodes in the network, there exists at most mc partitions in each level. Since partition cost is $\mathcal{O}(\log p)$ for a partition of size p, then the total cost can be loosely bounded by:

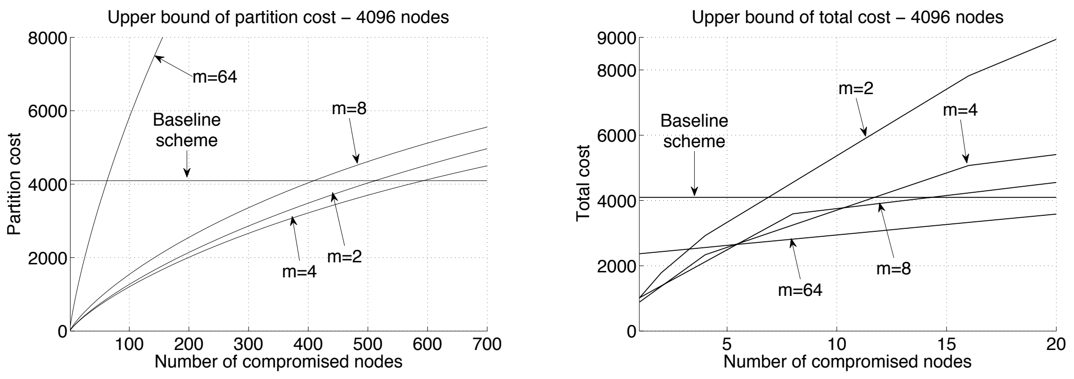
$$\begin{aligned} mc\sum_{i=0}^{\log n}\log\frac{n}{m^i} &= mc\left(\sum_{i=0}^{\log n}\log n - \sum_{i=0}^{\log n}\log m^i\right) \\ &= mc\left(\log^2 n - \log m\sum_{i=0}^{\log n} i\right) \\ &= mc(\log^2 n - \frac{\log m \log n}{2}(\log n + 1) = O(c\log^2 n) \end{aligned}$$

Since according to Theorem 3.5.1 partition cost of n nodes is $\mathcal{O}(c\log n)$, then total cost is bounded by $\mathcal{O}(c^2\log^3 n)$. □

3.5.3 A Numerical Example

To gain a better intuition of the results, we compute the cost associated with handling an adversary attack in a network of 4096 nodes (where c nodes are compromised) and analyze the graceful degradation of the network service. For test cost, we use the cost derived by [CPS06] as the more efficient bound of [FD08] is not a fixed cost characteristic.

Fig. 3.3(a) and 3.3(b) graph the maximum partition cost and total cost of the m-ary identification scheme, for different m. The baseline scheme is used in the graphs as a lower bound for when the proposed identification scheme is effective and efficient. Fig. 3.3(a) verifies the intuition that for a fixed number of corrupted

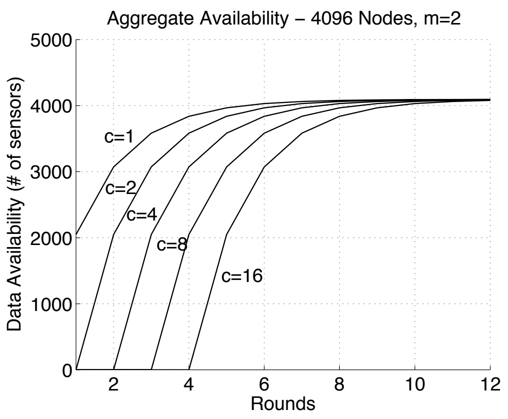


(a) Partition Cost (b) Total Cost

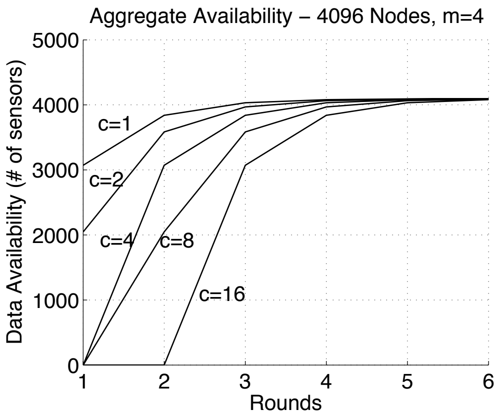
Figure 3.3: Cost of the identification scheme, as a function of partition degree.

nodes in the network c, the number of generated partitions increases with the partition degree m. This increase plateaus when the the identification scheme needs to test every single vertex on the identification tree. The performance of the m-ary identification scheme is best shown in Fig. 3.3(b) where the link cost for different m is compared with the baseline. It is clear that to optimize total cost, a network administrator choose an appropriate partition degree depending on probability of attack, vulnerability of the network as well as the necessary rapidity of the response (as response time is $\mathcal{O}(\log_m n)$). Fig. 3.3(b) also shows that test cost is the dominant term in total cost. This is promising as test cost is only dependent on the cost of the aggregation-verification primitive. More efficient primitives yield better results.

Fig. 3.4(b) and 3.4(b) shows the rate of improvement of the network service over the course of identification. Data is normalized by only looking at the number of nodes that contribute to the aggregate in a particular round. The maximum available data for a network of size n with c corrupted nodes, is $n - c$. In particular



(a) m=2



(b) m=4

Figure 3.4: Aggregate availability, as a function of partition degree m.

we note that if we fix c, as m is reduced, data becomes available in later rounds. This is because in the worst case, the first c partitions generated are corrupt.

3.5.4 Lower Bound and Average of Partition Cost

Up to now, we have focused on the upper bound or worst case partition cost. To gain an understanding for the behavior of the algorithm in practice, it is important to also analyze the lower bound and average behavior of the partition cost.

3.5.4.1 Average Cost

Computing the average partition cost when the number of malicious nodes in the network is fixed leads to a state space size exponential in the number of nodes in the network. To see this, observe that the number of possible states in each round is equivalent to the classic combinatorial balls-in-the-buckets problem where the balls and the buckets correspond to the nodes and partitions in the network respectively;

furthermore there are two types of balls (good nodes and bad nodes). Then the total number of possible states can be computed by the product of the number of states in each round. This leads to a combinatorial explosion in the number of states and therefore is hard to compute exactly.

3.5.4.2 Lower Bound Cost

The lower bound of the partition cost can be obtained by minimizing the number of vertices in the detection tree T. According to the algorithm, every internal vertex of T must be associated with an impure set and there must exist exactly c impure leaves. Partition cost can be minimized when both the number of pure leaves and the internal vertices in T are minimized. Figure 3.5 presents two different distributions of 3 compromised nodes in the network which yield maximum and minimum partition cost.

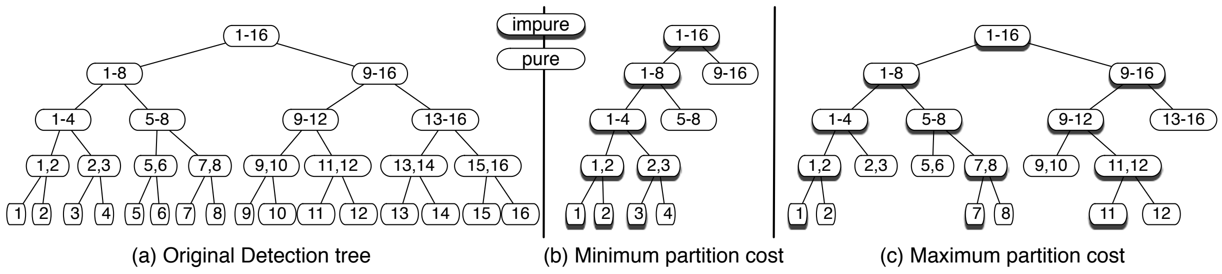
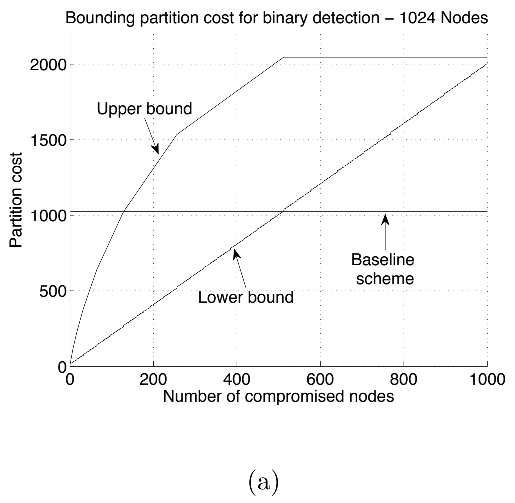


Figure 3.5: Partition cost depends on how the compromised nodes are distributed in the network.

Figure 3.6(a) graphs the lower and upper bounds of the partition cost of the identification algorithm in a network of 1024 nodes for different number of compromised nodes. As expected this results in much smaller cost than the worst case scenario and the cost increase is almost linear in the number of compromised nodes in the network. Indeed the intersection of the detection partition cost and the cost of the trivial scheme changes from 128 compromised nodes to 512. This means that the partition cost of the detection scheme is less than the total cost of the trivial scheme for up to 512 compromised nodes. Figure 3.6(b) presents the lower and upper bounds of the total cost of the identification algorithm. In the best case, the performance of the identification algorithm is better than the trivial case even up to 96 compromised nodes. However in the worst case, the scheme performs worst than trivial scheme when simply 3 nodes are compromised.



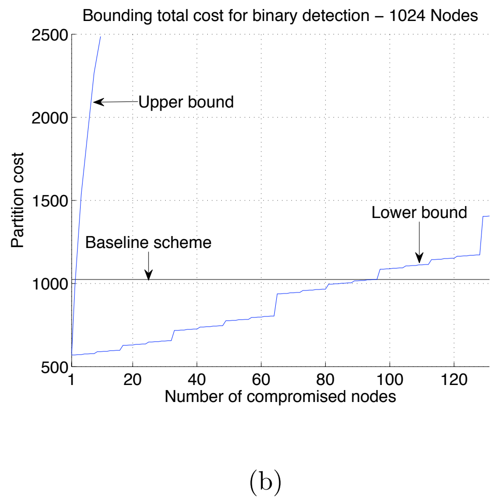


Figure 3.6: Bounding (a) partition cost and (b) total cost for binary identification algorithm.

3.5.5 Rational Adversary: Bounded Presence

We have clearly shown that the cost of the detection scheme is directly related to the distribution of the compromised nodes on the aggregation tree. Detection cost increases as nodes are more uniformly distributed along the aggregation tree. In particular, detection cost is maximized when the c compromised nodes are distributed such that they generate c disjoint subsets in as *early* a round as possible[2]. In contrast, cost is minimized when the compromised nodes are distributed so that they form a contiguous cluster and therefore generate the c disjoint subsets in as *late* a round as possible.

The intuition is evident as follows. Let nodes be identified according to their geographical locations on a one dimensional axis, such that nodes u_1 is neighbor to u_2, which is in turn neighbor to u_3 and so on. Cost is minimized when an adversary compromises a contiguous cluster of nodes $u_1, \cdots, u_c$. Of course not every clustering of nodes results in optimal cost since a cluster can span multiple detection subtrees. For example in Figure 3.5 cost of detection would be distinctly different if instead of compromising nodes u_1, u_2, u_3, which results in minimum partition cost, the cluster of nodes u_7, u_8, u_9 were compromised. Therefore, although compromising a cluster of nodes does not always result in minimum cost, it does save significant cost.

The above observation is indeed promising as a rational adversary compromises nodes that minimize its cost and maximize its benefit. It is rational behavior for an adversary to compromise a cluster of neighboring nodes (i.e., have a bounded presence) than to uniformly compromise nodes across the whole network. Compromising uniformly requires not only a greater deal of network access, but also increased risk of physical detection. We can therefore conclude that in the event of an attack, it is more likely for the detection cost to be closer to the lower bound

[2]This round number corresponds to the value of parameter v in Theorem 3.5.2.

(best case) than the upper bound (worst case).

3.6 Conclusion

Adversary attacks against data aggregation in ad hoc networks can have disastrous results, whereby a single corrupted node can affect the perceived measurements of large portions of the network by the BS. Current approaches to handling such attacks either aim exclusively at the detection of attack or provide inefficient ways of identifying corrupted nodes in the network, with respect to the baseline scheme; i.e., it is more efficient if sensor data is not aggregated at all. In this work, we presented a group-based approach to handling adversary attacks in aggregation applications, that identifies corrupted nodes while ensuring continuous, but gracefully degraded service during the attack period. Our analysis results in a precise cost-base characterization of when in-network aggregation retains its assumed benefits in a sensor network operating under persistent attacks. Our scheme is most effective when the adversary has corrupted a small fraction of the nodes in the network.

Although our work provides promising results in divide-and-conquer handling of attacks in aggregation applications, we have assumed a simplified adversary model. In the future, we plan on generalizing our model to account for non-persistent adversaries as well as allowing for identification error to decrease identification efficiency.

Chapter 4

Privacy-preserving Integrity-assured Data Aggregation

4.1 Single Aggregator Model

We consider the setting where n sensors are deployed in some area, remotely from a user $\mathcal{U}$. Sensors monitor and measure their environments and respond to the statistical queries of the user. An aggregator node $\mathcal{A}$ is used as an intermediary between the user and the sensor nodes that aggregates the sensor data and forwards the query response to the user. The aggregator can be thought of as a third party that computes an aggregation function f on the input sensor data $x_1, \ldots, x_n$ where x_i is the measurement of sensor s_i. The aggregator forwards the aggregate $y = f(x_1, \cdots, x_n)$ to the user $\mathcal{U}$.

We assume that all nodes have direct access to the aggregator node. All messages exchanged between the sensor nodes and the user, pass through the aggregator.

4.2 Privacy

Privacy in the single aggregator model can be preserved with respect to either an untrusted external user, aggregator or mutually distrustful sensor nodes. Privacy issues can range from individual measurements and transactional privacy to node identification, count, location or even sensing scheduling. In this work, we focus only

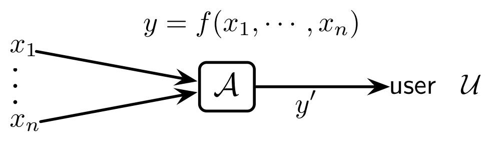


Figure 4.1: Single aggregator model

on preserving the privacy of *measured data* with respect to the *user*. Henceforth, our use of the term 'privacy' implicitly implies 'data sensed by individual nodes'.

A spectrum of privacy goals can be defined based on the controlled disclosure of private data. This controlled revelation may become an acceptable trade-off given the different efficiency and confidence requirements of a system as well as different aggregation function to be computed.

In this work, we define two distinct privacy goals. **Perfect privacy** is achieved when the adversary does not obtain any information about $\{x_i\}$ other than what it can deduce from the aggregate.

Distribution privacy is achieved when the adversary does not learn any information about the distribution of the data $\{x_i\}$, other than what it can deduce from the aggregate.

For clarity of definition, consider a sensor network where the environment to be sensed follows a probability distribution function $h(x)$. Let x_i be the sensed data of sensor s_i. To achieve perfect privacy, other than what user $\mathcal{U}$ can learn from the aggregate, $\mathcal{U}$ should not learn x_i; to achieve distribution privacy, $\mathcal{U}$ should not learn h.

Distribution privacy allows the protection of sensed data against attacks such

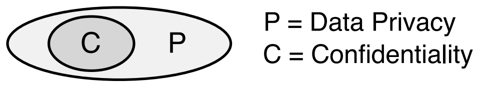


Figure 4.2: Relationship between confidentiality and data privacy

as tight estimation attack [AKSX04], where the aim of the adversary is to successfully determine a tight estimation of a secret plaintext, given its ciphertext. The tight estimation attack is clearly a relaxation of the security requirements of the ciphertext only attack (COA), where the aim of the adversary is to determine *any* information about a challenged ciphertext.

4.2.1 Privacy vs. Confidentiality.

In the context of aggregation applications, data confidentiality and data privacy, although related, are two distinct notions; viz., Figure 4.2. Privacy addresses not just data confidentiality but also hides any other information that can be revealed about the data, such as its distribution or various statistics. In contrast, confidentiality only deals with achieving perfect hiding of the data items and therefore can be considered to be a building block used to preserve privacy.

4.3 PIA Security Models

PIA deals with two distinct problems: privacy-preservation and aggregation integrity-assurance. Each problem is from the viewpoint of a different entity in the system and therefore has a separate adversary model and associated security goal that must be achieved.

4.3.1 Integrity-Verification

In the aggregation integrity-assured model, the user $\mathcal{U}$ wishes to verify the integrity of the aggregation process performed by an untrusted aggregator $\mathcal{A}$.

Assume a polynomially bounded integrity-adversary that can corrupt the aggregator node $\mathcal{A}$. The integrity-adversary can completely control the actions of $\mathcal{A}$ and learn any information residing on it.[1] The integrity-adversary's goal is to make the user accept a false aggregation result in a stealthy manner; i.e., not be detected by the user. The security goal of the user is to prevent stealthy attacks and ensure that the reported aggregate y' is "close enough" to the true aggregation value y—the accuracy of the aggregate depends on the verification efficiency we want to achieve.

Definition 4.3.1 *An aggregation scheme is perfectly secure if the adversary is unable to induce the user to accept an incorrect aggregation result.*

This definition is a stricter than the security definition provided by Chan et al. [CPS06] as in our scenario, sensors are trusted. Chan et al.'s definition accounts for untrusted sensors by allowing the presence of direct injection attacks—a direct data injection attack occurs when a malicious sensor node modifies the measured data it reports.

Approximation security was proposed by Pryzdatek et al. [PSP03]:

[1] In a more general model, the adversary can also corrupt a fraction of the sensors in the network and falsify the measurement value they report. Detection of such misbehavior however requires prior knowledge of the data distribution or applications/semantic specific knowledge. In this work, we don't address this type of attack since we do not make any assumptions on prior knowledge information.

Definition 4.3.2 *An aggregation scheme is (ϵ, δ)-approximation secure if the reported aggregate y' is bound by $(1-\epsilon)y \leq y' \leq (1+\epsilon)y$ and the probability that a malicious aggregator is not detected is upper bounded by δ.*

4.3.2 Privacy Preservation

In the privacy-preserving model, sensor nodes wish to preserve the privacy of their information with respect to the untrusted external user $\mathcal{U}$.

Assume a polynomially bounded privacy-preserving adversary that can corrupt user $\mathcal{U}$. The privacy-adversary can completely control the actions of the user and learn any information residing on it. Similar to the integrity-adversary, we do not bound the actions of the privacy-adversary, but limit the computational power and time of the privacy-adversary (polynomial in the security parameter). The goal of the attacker is to learn more about the measured data $\{x_i\}$ of the sensors $\{s_i\}$ than what is specified by the privacy goal.

Definition 4.3.3 *An aggregation scheme is private if $\mathcal{U}$ does not learn any information about the measured data $\{x_i\}$ defined by the privacy goal, other than what it can deduce from the aggregate y.*

4.4 Aggregation Functions

We define the following categories of aggregation functions: decomposable and comparison-based.

A *decomposable* function $f(x_1, \dots, x_n)$ can be expressed as:

$$f(x_1, \cdots, x_n) = \phi(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n))$$

where $\phi, g_1, \dots, g_m$ are some other functions. We say f is decomposed into functions $g_1, \dots, g_m$. Such functions are intrinsically hierarchical.

A *comparison-based* function $f(x_1, \cdots, x_n)$ can be computed by an algorithm which only uses the comparison operations greater-than and less-than. Quantile functions are instances of comparison-based functions.

4.5 PIA Solution 1

In the first PIA solution, we show how perfect data privacy can be achieved for a general class of functions in the centralized integrity verification model. Our solution uses homomorphic encryption to hide the data.

An encryption scheme $\mathcal{E}$ is *homomorphic* if it allows meaningful manipulation of ciphertexts; i.e., by performing a specific algebraic operation $\odot$ on the ciphertext, one can perform operation $\otimes$ on the plaintext; i.e., $\mathcal{E}(x_1) \odot \mathcal{E}(x_2) = \mathcal{E}(x_1 \otimes x_2)$.

Such capability seems a natural fit for data aggregation as it allows an aggregator to operate on encrypted data. Depending on the desired aggregation function, an appropriate homomorphic encryption scheme can be selected.

Existing schemes e.g., [CMT05] however, are not resilient against adversaries that attack the integrity of the data as homomorphic encryptions are malleable by design. Our proposed scheme combines homomorphism and message authentication codes (MAC) to construct an authenticated encryption scheme for the aggregator model. Note that to obtain a secure authenticated encryption scheme, the order in which the two primitives are composed is crucial.

In the following, we consider the sum aggregation function, which allows the computation of additive functions such as mean and standard deviation. This requires an encryption scheme which is homomorphic over the addition operation. This approach can be easily extended to other homomorphic operations, such as multiplication.

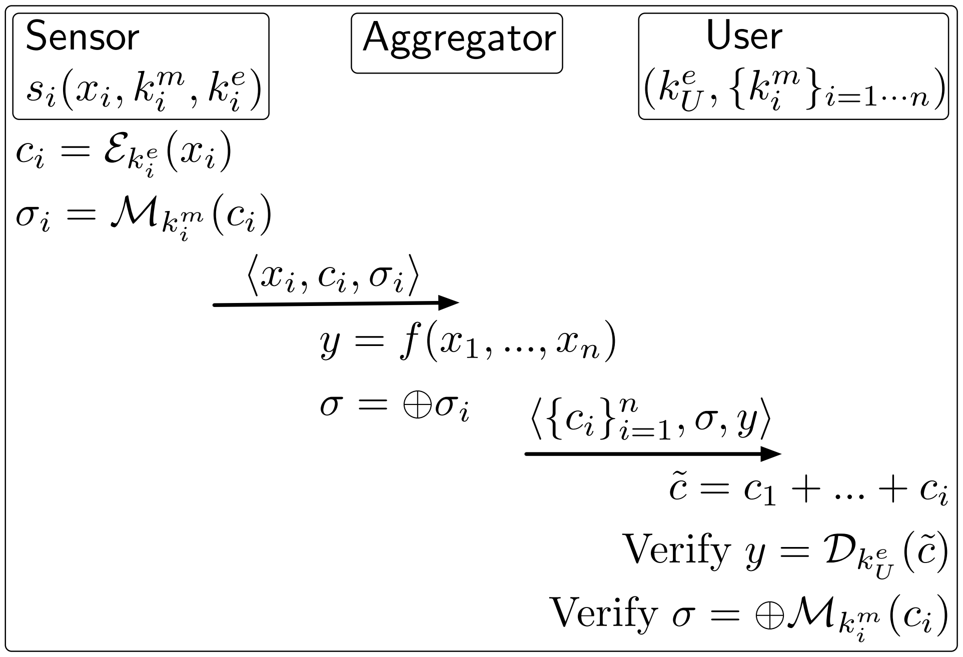
4.5.1 Assumptions.

Let $(\mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a perfectly private, symmetric-key additively homomorphic encryption scheme where $\mathcal{K}_e$ is the key generation algorithm, $\mathcal{E}_k(\cdot)$ is the encryption algorithm using key k and $\mathcal{D}_k(\cdot)$ is the decryption algorithm using k. An example scheme is proposed by Castelluccia et al. [CMT05]

We also assume the existence of a MAC protocol $(\mathcal{K}_m, \mathcal{M}, \mathcal{V})$ secure against existential forgeries in a chosen message attack, where $\mathcal{K}_m$ is the key generation algorithm, $\mathcal{M}_k(\cdot)$ is the tag generation algorithm using key k and $\mathcal{V}_k(\cdot)$ is the tag verification algorithm using k. An example instance of a secure MAC protocol is the HMAC [BCK96].

4.5.2 Protocol Description.

In the setup phase, a trusted third party generates and distributes an encryption key k_i^e and a MAC key k_i^m to each sensor node s_i. The user is given the



Sensor
$s_i(x_i, k_i^m, k_i^e)$
Aggregator
User
$(k_U^e, \{k_i^m\}_{i=1\cdots n})$
$c_i = \mathcal{E}_{k_i^e}(x_i)$
$\sigma_i = \mathcal{M}_{k_i^m}(c_i)$
$\langle x_i, c_i, \sigma_i \rangle$
$y = f(x_1, ..., x_n)$
$\sigma = \oplus \sigma_i$
$\langle \{c_i\}_{i=1}^n, \sigma, y \rangle$
$\tilde{c} = c_1 + ... + c_i$
Verify $y = \mathcal{D}_{k_U^e}(\tilde{c})$
Verify $\sigma = \oplus \mathcal{M}_{k_i^m}(c_i)$

Figure 4.3: PIA Solution 1

encryption master key $k_U^e = \sum_{i=1}^n k_i$ and also MAC keys k_i^m for $i = 1, \ldots, n$. Note that the user cannot deduce any information about the individual sensor encryption keys using k_U^e.

Sensor s_i encrypts its data x_i and computes a MAC tag σ_i over c_i and forwards them to the aggregator. The aggregator computes the data aggregate y and the MAC aggregate using the XOR function and forwards $\langle \{c_i\}_{i=1}^n, \sigma, y \rangle$ to user. The user verifies the correctness of the alleged aggregate y by decrypting the sum of the ciphertexts $\{c_i\}_{i=1}^n$ using k_U^e. If the aggregate verifies, it verifies the aggregate MAC tag to ensure the aggregator has not changed the data ciphertexts.

4.5.3 Analysis

Theorem 4.5.1 *PIA solution 1 is perfectly private.*

Given that the encryption scheme is perfectly private, the user cannot learn any information about raw data x_i from the individual ciphertexts c_i, except what is revealed by the aggregate, which in this case is simply the sum of the ciphertexts.

Next we show security of the system against the integrity-adversary. Intuitively, the scheme is secure because the composition of a perfectly private encryption scheme and an existentially unforgeable MAC scheme results in an authenticated encryption scheme which is INT-PTXT ("integrity of plaintext") secure.

Theorem 4.5.2 *PIA solution 1 is perfectly secure.*

Proof: We need to show that an integrity-adversary cannot induce the user $\mathcal{U}$ to accept an incorrect aggregation result y' such that $y' \neq \sum_{i=0}^{n} x_i$. According to the protocol, $\mathcal{U}$ accepts y' if first, ciphertext $\tilde{c} = \sum_{i=1}^{n} c_i$ decrypts to y' and second, the aggregate tag σ verifies correctly. Since semantic security of the encryption scheme is only intended against a passive adversary, the integrity-adversary has the capability of modifying ciphertext c_i without being detected. However the integrity-adversary can induce $\mathcal{U}$ to accept the second test if and only if it can forge a valid MAC tag for c_i. Assuming that the MAC protocol is existentially unforgeable, then the integrity-adversary can succeed at best, with negligible probability. $\square$

4.5.4 Discussion

Although a homomorphic based scheme is perfectly private, the approach limits the aggregation functions that can be supported—in particular, only functions that the encryption scheme is homomorphic over are supported. An additive homomorphic scheme supports functions such as add, average and standard deviation. All other functions are not supported, including the large set of comparison-based functions, such as min/max or median. This is because an encryption scheme which is perfectly secure or COA-secure cannot be homomorphic over a comparison function [RAD78] as the adversary can learn information about the order of the ciphertexts and thus break the scheme.

An indirect effect of the scheme's inability to support comparison operators is that a range bound cannot be placed on the measured data, in order to bound the effect of malicious sensors.

Finally, the scheme is highly efficient given the resource constrained sensor nodes. In fact, each sensor sends the aggregator three values. The bulk of the communication $\mathcal{O}(n)$, is between the aggregator and the user, which generally have improved resources. However for large n, the scheme can still be inefficient.

4.6 PIA Solution 2

PIA Solution 1 shows that perfect privacy can be achieved in the centralized integrity verification model but at the expense of limiting the aggregation functions that are supported.

In PIA Solution 2, we show a tradeoff between measure of privacy and the type of aggregation function that the integrity verification can support. In particular, we show how, in the centralized integrity verification model, we can support comparison operator and quantile functions. The k-quantile of a data set is the point at which k fraction of the values in the set are below a given point. For instance, the 0.5-quantile (also called the median) is the point at which 50% of the data fall below.

Quantiles functions are especially useful in adversarial environments where the attacker can compromise sensor nodes and transmit false data values. Detection of such attacks, without prior knowledge of the data distribution, is very hard. Recently, Wagner [Wag04] showed that the median of a data set is more resilient in adversarial environments than the mean because unlike the mean, an adversary cannot freely dictate the median of the data set.

Recall from Section 4.5 that any perfectly private encryption scheme used to hide the data from the user excludes the support of comparison-based aggregation functions, such as the quantile function. However we observe that by relaxing our notion of privacy we can support comparison-based aggregation functions. In particular, by adopting an order preserving encryption scheme (OPES) we can preserve the privacy of the distribution of the data—OPES preserves the order of the data; i.e., any pair of plaintexts x_1, x_2, where $x_1 < x_2$, encrypt to ciphertexts c_1, c_2, such that $c_1 < c_2$.

PIA Solution 2 uses an OPES scheme to verify the integrity of comparison-based aggregation functions. The scheme, shown in Figure 4.4, works in three stages:

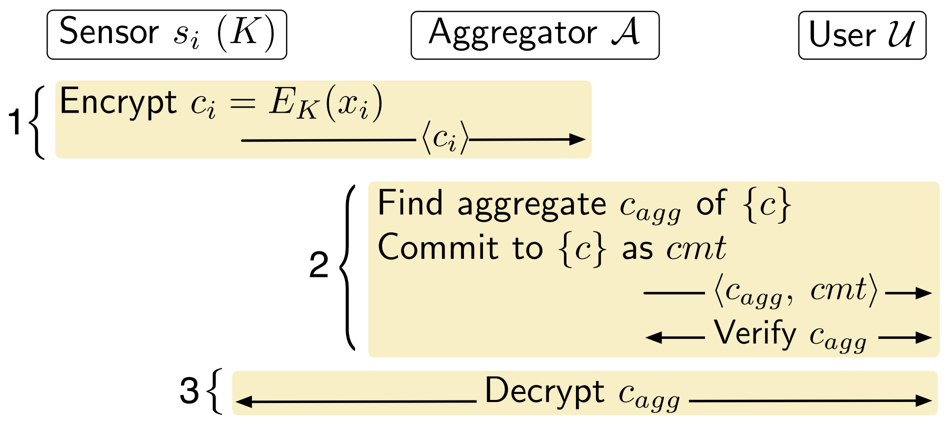


Figure 4.4: Overview of PIA Solution 2

1. Encrypt: Sensors encrypt their data using an OPES with a master secret key shared by all the nodes in the network. The encrypted data set C is sent to the aggregator.

2. Aggregation-Verification: The aggregator determines the aggregate c_{agg} of the encrypted data set C and engages the user in a protocol, to verify the correctness of the alleged encrypted aggregate c_{agg}.

3. Decrypt: The user queries the sensors to decrypt c_{agg} and find the plaintext of the data aggregate.

4.6.1 Protocol Description

4.6.1.1 Encryption

We need an encryption scheme which takes as input a target distribution and transform the plaintext values in such a way that the transformation preserves the order while the transformed values follow a target distribution. In addition, our

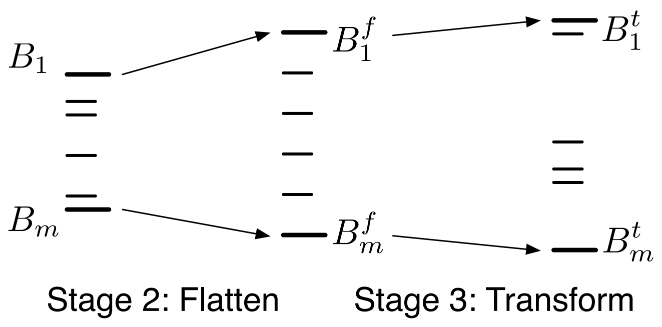


Figure 4.5: Bucket distribution transformation in OPES

adversary can have access to all ciphertext values and does not have any other information, including any special information about the domain of the plaintext data. This closely corresponds to the scenario of a remotely deployed sensor network and a user querying the sensed data.

We use the OPES scheme of Agrawal et al. [AKSX04] proposed originally for database encryption. The scheme only uses addition and multiplication operations and has been shown to be practical for MICA2 motes [AGW05].

OPES works in three stages. We briefly outline each stage below and refer the reader to [AKSX04] for the details.

1. Model. Distinct data values from both the input and the target distributions are first partitioned into buckets that capture the statistical information about the distributions. The distribution in each bucket is then modeled as a linear spline, where the spline for a bucket $[x_l, x_h)$ is simply the line connecting the densities at the two end points of the bucket.

2. Flatten. The plaintext data set X is transformed into a flat data set F

such that the values in F are uniformly distributed.

3. Transform The flat database F is transformed into the cipher data set C such that the values in C are distributed according the target distribution. The target distribution is chosen so that the adversary is forced to make large estimation errors.

For further details of OPES, we refer the reader to [AKSX04].

4.6.1.2 Aggregation-Verification

Once the data is encrypted and forwarded to the aggregator $\mathcal{A}$ who aggregates the data, the user $\mathcal{U}$ initiates an aggregate integrity verification algorithm. There are two approaches to verify the integrity of an alleged aggregate y, differing in the verification accuracy that they provide.

If a user desires perfect verification (e.g., in [DDHV03]), then the user requires access to all the raw sensor data $x_1, \cdots, x_n$ in order to recompute and check if $y \overset{?}{=} f(x_1, \cdots, x_n)$. In this case the aggregation integrity verification algorithm is straightforward: the aggregator simply forwards the user an alleged ciphertext aggregate c_{agg} and the complete set of data ciphertexts $c_1, \ldots, c_n$. The user can thus verify the correctness of c_{agg} for any comparison-based aggregation function using the individual data ciphertexts. However, in some scenarios, the high communication overhead between the aggregator $\mathcal{A}$ and the user $\mathcal{U}$ (linear in the number of sensors in the network) might not be desirable. This is in particular the case, if the network is large.

To reduce the communication between $\mathcal{A}$ and $\mathcal{U}$, Przydatek et al. [PSP03] propose an aggregate-commit-prove framework where a user samples a subset of the raw data and computes an approximation of the aggregate to ensure that the alleged aggregate is "close enough" to the approximation. This framework can be used to verify any function that can be approximated by *uniform sampling* of data.

For our purposes however, we use the set of aggregation-verification schemes (called SIA) of Przydatek et al. [PSP03] as they improve the proof efficiency of functions such as quantile, median and min/max. SIA offers a tradeoff between the accuracy of the approximation and the communication cost between $\mathcal{A}$ and $\mathcal{U}$. We refer the reader to [PSP03] for the details. In the following, we briefly describe the framework, with special attention to the median function.

Upon receiving the encrypted data c_i from each sensor s_i over an authenticated channel, $\mathcal{A}$ first computes the aggregate and then forms a commitment over the collected data using a Merkle hash tree. This forces $\mathcal{A}$ to choose a fixed set of data values. The aggregator then forwards the aggregate and the commitment to $\mathcal{U}$. The user then engages the aggregator in a proof protocol where it verifies that $\mathcal{A}$ has used the data provided by the sensors and that the aggregate provided is close enough to the true aggregate.

As an example, we describe the interactive proof $\mathcal{U}$ initiates with $\mathcal{A}$ to be assured of the integrity of an alleged median value. Let $\mathcal{A}$ and $\mathcal{U}$ communicate over authenticated channels.

Algorithm 4 *MedianCheck [PSP03]*

Input: *network size* n*, alleged aggregate* y*, commitment* h*,* ϵ*-approximation accuracy*

1. $\mathcal{U}$ *verifies that the committed sequence is sorted and that all the elements in the sequence are distinct. This is done by selecting two elements* x_i, x_j *from random positions in* S *and ensuring that* $x_i < x_j$ *if* $i < j$*.*

2. $\mathcal{U}$ *verifies that* y *is (close to) the median of the committed sequence. Here, the user selects elements from random positions in the committed sequence and ensures that the elements picked from the first and second half of the sequence are respectively smaller and bigger than* y*.*

Output: *If both tests are valid, Accept.*
Else, with probability at least $(1 - 1/e) > 1/2$*, Reject.*

4.6.1.3 Decryption

Up to now, the user has verified the correctness of the encrypted aggregate c alleged by the aggregator. If c is incorrect, the user terminates the protocol; else if c is correct, the user decrypts the value by querying the appropriate sensor to obtain the corresponding aggregate plaintext.

4.6.2 Analysis

Theorem 4.6.1 *PIA Solution 2 achieves distribution privacy.*

Proof: In addition to the aggregate value, user $\mathcal{U}$ sees only encrypted data. $\mathcal{U}$ obtains a set of ciphertext values during the aggregate-verification stage. However, since $\mathcal{U}$ does not have access to the decryption secret key and the original OPES scheme perfectly hides the distribution of the sensor data (for proof refer to [AKSX04]), then $\mathcal{U}$ cannot learn any information about the distribution of the original data. $\square$

Theorem 4.6.2 *In PIA Solution 2, the user can verify that the alleged aggregate is the ϵ-approximation of the median with probability at least $1 - 1/e$, by requesting $\mathcal{O}(\log n/\epsilon)$ elements from the aggregator.*

Proof: Once the sensor nodes submit their encrypted data to the aggregator $\mathcal{A}$ and $\mathcal{A}$ commits to the data by constructing a Merkle hash tree, the aggregator cannot later change the ciphertext values. This is because of the collision-resistant property of the hash function used in the Merkle hash tree. We use Theorem 2 from [PSP03] to prove that the user can verify with said probability that the alleged ciphertext median is ϵ-approximation of the median of the ciphertext data set C. This is because the encryption scheme preserves the order of the data. Once this is verified, $\mathcal{U}$ queries the appropriate sensor to decrypt their ciphertext. Since sensor nodes are honest, $\mathcal{U}$ obtains the valid median plaintext. $\square$

Lemma 4.6.1 *PIA Solution 2 supports comparison-based aggregation functions and requires $\mathcal{O}(1)$ communication per sensor.*

Although we described the integrity verification scheme for the median function, we can easily extend the scheme for k-quantile type functions by changing the k-separating of the committed sequence. In general, this framework is useful for any aggregation function which can be approximated by uniform sampling [PSP03, BYKS01] and which relies only on the comparison operation.

4.7 PIA Solution 3

Next we focus on the *distributed* integrity verification model (refer to Figure 4.6) and show how we can achieve perfect privacy and integrity assurance. Recall that this model transfers the recomputation of the aggregate to the sensor nodes. Consequently, the solution is intrinsically privacy preserving as the user never needs to access the raw data; i.e., the solution is private without requiring any additional privacy preserving mechanisms such as encryption. However as we will show, this privacy comes at the expense of increases communication between the sensor nodes and depending on the application, this communication might neither be desired or even feasible.

The distributed integrity verification approach was first proposed by Chan et al. [CPS06] for in-network aggregation. In their proposed Secure Hierarchical In-network Aggregation (SHIA) scheme, a commitment is constructed over the aggregation process, forcing the integrity-adversary to choose a fixed aggregation topology and set of aggregation results. Once the aggregation process is complete, each sensor independently reconstructs the commitment tree and ensures that the adversary has not modified or discarded the contributions of that node.

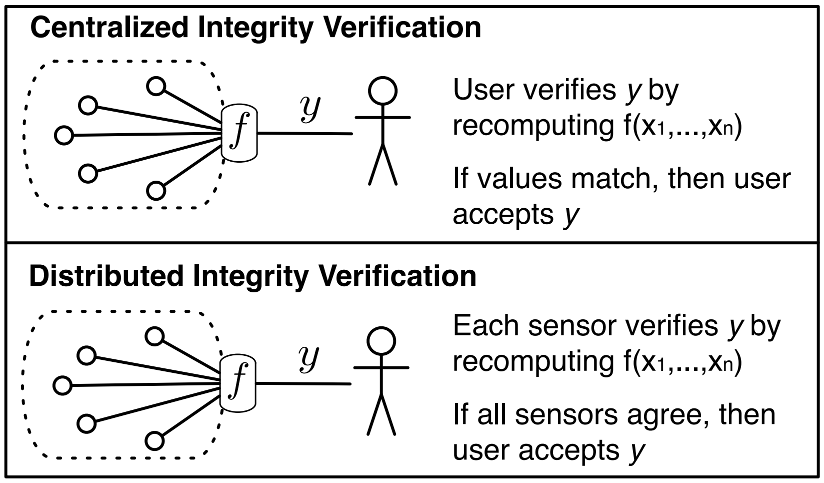

Centralized Integrity Verification
f
y
User verifies y by recomputing f(x1,...,xn)
If values match, then user accepts y
Distributed Integrity Verification
f
y
Each sensor verifies y by recomputing f(x1,...,xn)
If all sensors agree, then user accepts y

Figure 4.6: The difference between centralized and distributed aggregation integrity verification models.

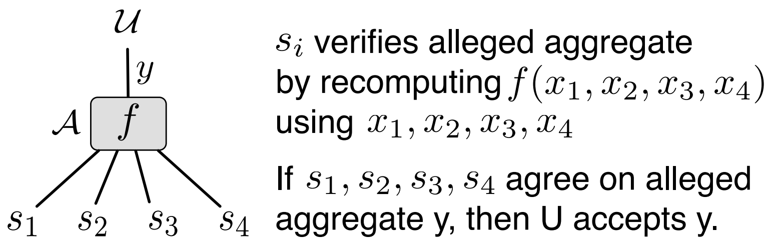

$\mathcal{U}$
y
$\mathcal{A}$
f
s_1
s_2
s_3
s_4
s_i verifies alleged aggregate by recomputing $f(x_1, x_2, x_3, x_4)$ using x_1, x_2, x_3, x_4
If s_1, s_2, s_3, s_4 agree on alleged aggregate y, then U accepts y.

Figure 4.7: PIA Solution 3: Basic distributed integrity verification, where x_i is the data of sensor s_i

4.7.1 Basic PIA Solution 3.1

We adapt distributed integrity verification from the in-network aggregation model to the single aggregator model. We refer to the direct adaptation of SHIA as the basic solution, as seen in Figure 6.

Algorithm 5 *Basic PIA Solution 3.1*

Let k_i be a MAC key shared between s_i and $\mathcal{U}$. Let H denote a secure cryptographic *hash function.*

1. $\mathcal{A}$ computes $h = H(y, s_1, x_1, \ldots, s_n, x_n)$ and sends $\langle y, h\rangle$ to $\mathcal{U}$, where y is the alleged aggregate.

2. $\mathcal{U}$ sends an authenticated broadcast $\langle h\rangle$ to all sensors s_i.

3. s_i gets from $\mathcal{A}$ pairs (s_j, x_j), $\forall j \neq i$. s_i computes aggregation function $y' = f(x_1, \ldots, x_n)$ and commitment h_i using the data pairs. If $h_i = h$, then s_i sends $c_i = H(k_i, OK)$ to $\mathcal{U}$.

4. If $\mathcal{U}$ verifies that all sensors s_i agree with the value h, then the alleged aggregate y is correct.

Theorem 4.7.1 *Basic PIA Solution 3.1 is perfectly private and perfectly secure. The scheme can support any aggregation function, at the cost of $\mathcal{O}(n)$ messages per sensor.*

Proof: The security of the basic scheme against an integrity-adversary can be reduced to the security of SHIA proven in [CPS06]. Additionally, given that the hash function is one way, $\mathcal{U}$ doesn't get any information on the sensor measurements, thus scheme is perfectly private.

The scheme can support any aggregation function since sensors have access to all the raw data. However, this means that each node receives $\mathcal{O}(n)$ messages during the integrity verification algorithm. $\square$

Remark. The high communication cost of the scheme eliminates the assumed efficiency benefits of aggregation.

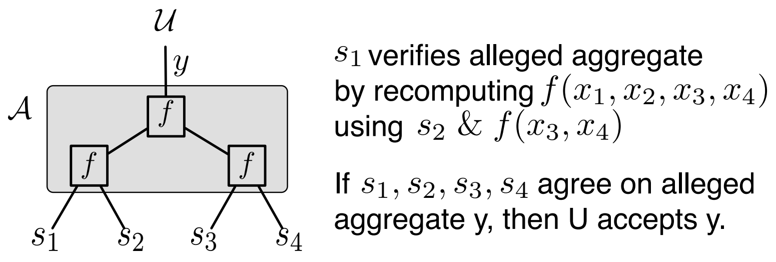

$\mathcal{U}$
y
$\mathcal{A}$
f
f
f
s_1
s_2
s_3
s_4
s_1 verifies alleged aggregate by recomputing $f(x_1, x_2, x_3, x_4)$ using s_2 & $f(x_3, x_4)$
If s_1, s_2, s_3, s_4 agree on alleged aggregate y, then U accepts y.

Figure 4.8: PIA Solution 4: Improving efficiency by introducing a logical aggregation tree within the aggregator node.

4.7.2 Improved PIA Solution 3.2

We now propose an algorithm that improves the efficiency of the integrity verification algorithm but at the cost of limiting the aggregation functions that can be supported. This is done by introducing a logical aggregation tree within the aggregator node as seen in Figure 4.8. Accordingly, to recompute the network aggregate, each node would require $\log n$ values instead of n values. We consider the aggregation function sum.

Algorithm 6 *Improved PIA Solution 3.2*

Let k_i be a MAC key shared between s_i and $\mathcal{U}$. Let H denote a secure cryptographic hash function.

1. $\mathcal{A}$ constructs a logical binary aggregation tree $\mathcal{T}$ where the leaf vertex v_i is associated with sensor s_i, with label $\ell_i = (1, x_i, s_i)$. The internal vertex v_i has label $\ell_i = (c_i, a_i, h_i) = (c_1 + c_2, a_1 + a_2, H(c_i, \ell_1, \ell_2))$ where $\ell_j = (c_j, a_j, h_j)$, $j = 1, 2$ are labels of the leaf vertices of v_i. Then $\mathcal{A}$ sends ℓ_k to $\mathcal{U}$ where v_k is the root of $\mathcal{T}$.

2. $\mathcal{U}$ sends an authenticated broadcast of h to all sensors s_i.

3. $\mathcal{A}$ sends s_i the labels of all siblings of the vertices that are on the path from s_i to the root of $\mathcal{T}$. This allows s_i to recompute the label of the root of $\mathcal{T}$ and verify the correctness of h. If h is correct, then s_i sends $c_i = H(k_i, OK)$ to $\mathcal{A}$ which aggregates the values using the xor operation before forwarding to $\mathcal{U}$.

4. If $\mathcal{U}$ verifies that all sensors s_i agree with the value h, then the alleged aggregate y is correct.

Theorem 4.7.2 *Improved PIA Solution 3.2 is perfectly private and perfectly secure. The scheme supports all decomposable aggregation functions at the communication cost of $\mathcal{O}(\log n)$ per sensor.*

Proof: The security proof of the algorithm closely follows that of SHIA [CPS06]. The proof outline is as follows: Commitment h of the root node of $\mathcal{T}$ forces the adversary to choose a fixed set of leaf vertices labels. If any internal vertex v_i of $\mathcal{T}$ does not compute sub-aggregate a_i correctly, then it will be detected by the sensors as either (1) the disseminated label doesn't match the computed one, or (2) h is not verified. User $\mathcal{U}$ can detect if any sensors does not agree with h as the HMAC tag can be forged with at best negligible probability.

The scheme is perfectly private as the hash function is secure and $\mathcal{U}$ doesn't get any other information about $\{x_i\}$.

The integrity verification algorithm requires each node s_i to recompute the sub-aggregates of the internal vertices on the path from s_i to the root node of $\mathcal{T}$. Since $\mathcal{T}$ has n leaf vertices, then the path length is $\log n$. Thus s_i receives $\mathcal{O}(\log n)$ messages.

Finally, the solution can only support decomposable functions which allow for the construction of the hierarchical aggregation tree $\mathcal{T}$. Such functions include, mean, standard deviation, count and min/max. $\square$

4.8 Conclusion

As large sensor networks become prevalent in everyday life, privacy becomes a critical issue that must be addressed. In this chapter, we point out the role of privacy in integrity-assured data aggregation. We define the problem and analyze the security model. We then investigate the tradeoff between privacy and aggregate integrity verification in the single aggregator model. Our results, summarized in Figure 1.5, show a clear tension between the privacy of the sensed data from the user and the cost of the integrity verification. This cost consists of both the communication incurred by sensor nodes as well as the range of aggregation functions that can be supported by the integrity verification algorithm.

Chapter 5

PIA in Publish-Subscribe Systems for Multiple Subscribers

In this chapter we address the problem of privacy-preserving integrity-assured aggregation in publish-subscribe systems for multiple subscribers, where subscribers can gain a competitive advantage if they can obtain verified aggregate information before the others. Our solution distributes the verification functionality so that the scheme satisfies the *fairness* property in the sense that subscribers can either all correctly verify the aggregation process or they can detect all cheating subscribers. We then formally analyze the security properties of our scheme against various attack models.

5.1 System model

5.1.1 Assumptions

We assume that a pub-sub system consists of a large set of publisher and routing nodes in a *routing network*, while a small set of subscribers exist outside of the routing network, as shown in Figure 5.1. When a subscriber subscribes to the aggregate sum of data maintained by a set of publishers, the pub-sub system constructs a routing tree along which routing nodes perform in-network aggregation. Exactly how the pub-sub system should compute this routing tree is out of the scope of this paper—our aggregation protocol is independent of the routing path

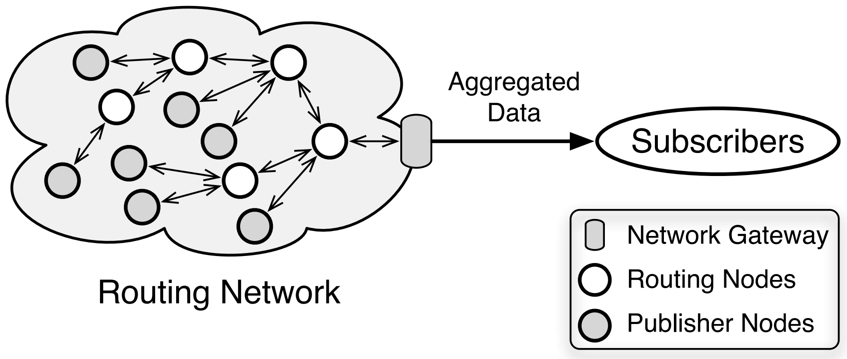


Figure 5.1: Publisher-subscriber model

computation algorithm.

After the routing path is established, each publisher p_i publishes a variable v_i periodically synchronizing with the other publishers; that is, all the publishers publish their variables of each round with a sequence number n. This data is forwarded along a routing tree where each routing node performs in-network data aggregation. Each routing node aggregates variables with the same sequence number and sends the aggregated variable to its parent node in the routing path. If each publisher p_i for $i = 1$ to n publishes a variable v_i, each subscriber eventually receives from the root routing node of the routing tree an aggregate sum $\sum_{i=1}^{n} v_i(n)$ at each round n.

Each entity, which manages either a publisher, a subscriber, or a routing node, has a unique identifier. Every pair of publisher p_i and subscriber s_j share a pairwise secret K_i^j. Note that publishers in our system model are more tightly coupled with subscribers than in many existing pub-sub systems; each subscriber explicitly specifies which publishers should provide raw data for the aggregate sum in a subscription request. We also assume that subscribers know the routing topology within the routing network. This is an acceptable assumption since publisher nodes are generally long lived and the routing topology is not dynamic.

Our aggregation protocol requires bidirectional communication channels among publishers and routing nodes inside the routing network. However, although the routing network forwards messages from routing nodes to subscribers, it does receive messages from those subscribers to either routing nodes or publishers.

5.1.2 Security Properties

In this paper, we study the problem of data privacy and integrity in a publisher-subscriber setting. A pub-sub system that supports in-network aggregation should satisfy the following security properties:

- **Data privacy:** Each subscriber learns only the aggregate of a set of published data and no other information about the individual published data.

  (We assume that publishers trust the routing nodes with respect to the privacy of their data.)

- **Aggregation integrity:** A subscriber can verify the correctness of the alleged aggregate without seeing raw data from the publishers.

- **Fairness:** Each subscriber can verify the integrity of the aggregate if and only if all the other subscribers who also subscribe to that aggregate function collaborate correctly.

  A malicious subscriber who deviates from the protocol is always detected and identified.

5.1.3 Security Model

We consider three different types of adversaries who try to violate the security properties in Section 5.1.2. We also define security of the system with respect to each adversary.

Privacy adversary (p-adversary) attacks the privacy of raw data from publishers. Such an adversary is typically a subscriber who resides outside of the routing network and thus cannot access any messages within the routing network except for those forwarded to the adversary from routing nodes in the network.

We consider that the system is secure against the p-adversary if at best, the adversary can learn only the aggregate and no other information about the individual published data.

Integrity adversary (i-adversary) attacks the security of the aggregation process. The integrity-adversary can compromise a number of routing nodes in an routing tree. Once a node is corrupted, the adversary has total control over the private data of that node as well as its subsequent behaviors. On the other hand, we assume that publishers are not part of an adversary against subscribers because publishers can always modify the aggregated data by providing malicious input data while otherwise following the aggregation protocol properly.

The objective of the adversary is to tamper with the aggregation process such that subscribers accept an aggregation result, which is not the correct aggregate of the publishers' raw data. An aggregation protocol is considered secure if the adversary cannot successfully launch such an attack.

Subscriber adversary (s-adversary) can compromise one or more subscriber nodes. Once compromised, the adversary has subsequent access to the subscriber's private information as well as control over its behaviors. The adversary is active, meaning it can deviate from the protocol arbitrarily. The objective of the s-adversary is to interfere with the collaborative verification of the aggregation process and convince the other subscribers to accept a false verification result or prevent them from accepting the correct aggregate.

The system is secure against the subscriber adversary, if at the end of the protocol, honest subscribers can either correctly verify the integrity of the aggregation process, or identify the malicious subscribers.

5.2 Basic Scheme

In this section we present our basic secure aggregation protocol for multiple non-collaborative subscribers for the SUM aggregate function. The subscribers do not interact with each other in the basic protocol. Our scheme allows multiple subscribers to verify the integrity of an aggregate sum independently, while ensuring that the individual published data remains confidential with respect to the subscribers. Note that the basic scheme described in this section does not satisfy the fairness property in Section 5.1.2 we require; we will extend the basic protocol in Section 5.3 to achieve that property.

The basic protocol builds based upon the secure hierarchical in-network aggregation (SHIA) scheme developed by Chan et al. [CPS06] for sensor networks. We adapt it to the publish-subscribe network model by eliminating all subscriber-initiated communication to the routing network. For the sake of brevity, we mainly describe the following differences from SHIA.

- Elimination of an authenticated broadcast message from a subscriber, which corresponds to a base station in a sensor network, to the nodes in the routing network.
- Support of multiple subscribers to the same sum.

We only give only a high level description (and omit the details) of sub-protocols of our scheme that are identical to those of SHIA. We refer the reader to the original paper [CPS06] for further detail.

5.2.1 Protocol Overview

The basic scheme relies on an aggregation-commit-prove framework of SHIA [CPS06]. Suppose that subscriber node s_i subscribes to the sum of the values from publishers $p_1, \ldots, p_n$ in set $\mathcal{P}$. As routing nodes aggregate data from publisher nodes, they also iteratively construct a structure, which forms a commitment tree over the published data using a cryptographic hash function. In particular, the publisher nodes and routing (aggregating) nodes respectively form the leaf and internal vertices of the commitment tree. The final commitment value is the value of the root node of the commitment tree. Once the aggregation and commitment phase is complete, each publisher receive the off-path values of the commitment tree from other routing nodes and recompute the final commitment value. Each publisher then forms a verification tag that summarizes its view of the aggregation process. A subscriber can verify that all the published data were correctly aggregated to the received sum by checking that the views of the aggregation process for all the publishers are identical.

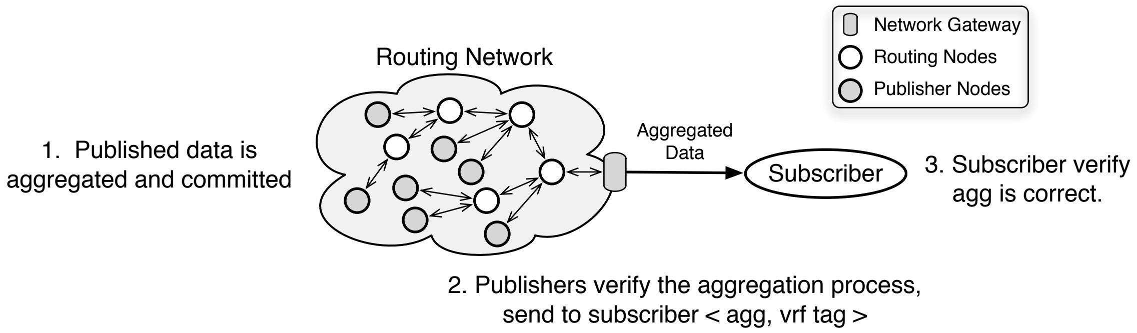

Network Gateway
Routing Nodes
Publisher Nodes
Routing Network
Aggregated Data
1. Published data is aggregated and committed
Subscriber
3. Subscriber verify agg is correct.
2. Publishers verify the aggregation process, send to subscriber < agg, vrf tag >

Figure 5.2: Overview of the basic scheme, for single subscriber.

The intuition of the protocol is that once the aggregation process is fixed, each publishers individually ensures that the its data was added to the final aggregate. This is done by rebuilding the aggregate and the commitment structure. If a malicious aggregator at some point attempts to discard or reduce a legitimate publisher's contribution, an inconsistent commitment structure will be generated and the attack will be detected. This approach ensures a lower bound to be established for the SUM aggregate. An upper bound is also established using a similar approach and the COMPLEMENT aggregate, defined as $\sum(r - d_i)$ where r is the upper bound of the valid data range and d_i is the data of publisher i.

The algorithm consists of the following two main stages: aggregation-commit and verification. We assume that an aggregation tree already exists and spans over the set of the publishers and routing nodes.

5.2.1.1 Aggregation-Commit

This phase is exactly same as that in SHIA. Each publisher node publishes data to its parent routing node on the routing tree. Once a routing node hears back from all of its children, it aggregates their data and constructs a new vertex label in the commitment tree[1]. The label has the following format:

$$\langle \mathsf{count}, \mathsf{value}, \mathsf{complement}, \mathsf{commitment} \rangle$$

where count is the number of leaf nodes in the subtree rooted at this routing node; 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 to value of the label and the labels of the child nodes in the commitment tree.

The labels are defined inductively as follows: There is one leaf vertex u_p for each publisher node p, which we call the leaf vertex of p. The label of u_p consists of count=1, value=d_p where d_p is the data value published by p, complement=$r - d_p$ where r is the upper bound on allowable data values, and commitment is the publisher's unique ID. Internal vertices represent the aggregation operations, and have labels that are defined based on the labels of their child vertices. Suppose an internal vertex has child vertices with the following labels: $\ell_1, \cdots, \ell_q$, where

[1]Note that the aggregation tree and the commitment tree are identical in our scheme.

$\ell_i = \langle c_i, v_i, \bar{v}_i, h_i \rangle$ for $i = 1$ to q. Then their parent vertex has label $\langle c, v, \bar{v}, h \rangle$ where $c = \sum c_i$, $v = \sum v_i$, $\bar{v} = \sum \bar{v}_i$ and $h = H[N, c, v, \bar{v}, \ell_1, \cdots, \ell_q]$. Here H is a cryptographic hash function and N is the session sequence number.

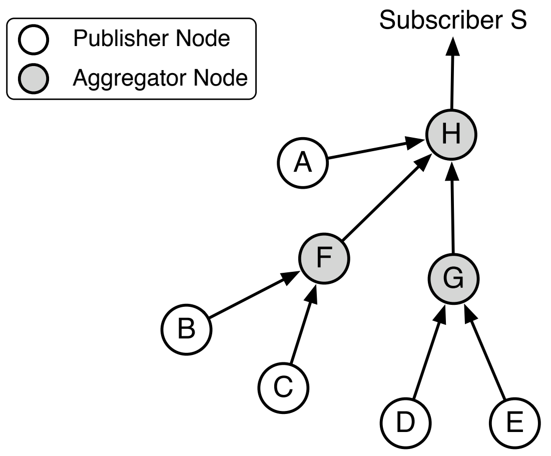
Once the label is constructed, the routing node forwards it to its parent routing node. Once the routing node at the root of the aggregation tree computes the final label, it can either forward that label to the subscriber or wait until it also has the result tags before forwarding.

5.2.1.2 Verification

In this stage, unlike SHIA, our protocol allows a subscriber to verify an aggregate without sending an authenticated broadcast message to the publisher nodes in the routing network. First, routing nodes disseminate information to the publisher nodes that allow them to verify that their published data was correctly incorporated into the aggregate. In particular, each publisher node p_j receives the labels corresponding to the set of all the siblings of each of the commitment tree vertices that are on the path from p_j to the root of the commitment tree. Once each publisher p_j reconstructs the label ℓ'_r of the root node r of the commitment tree, it computes the following verification tag for subscriber s_i:

$$\tau_j^i = H(K_j^i, N, \ell'_r)$$

where K_j^i is a secret key shared between publisher p_j and subscriber s_i. Notice that the label ℓ'_r computed by p_j could be different from the label ℓ_r computed at the aggregation-commit phase in Section 5.2.1.1 if there exist malicious routing nodes



Aggregation-commit stage:

$\ell_E = \langle 1, e, r - e, H(E) \rangle$
$\ell_G = \langle 2, g, \bar{g}, H(2, g, \bar{g}, \ell_D, \ell_E) \rangle$
$\ell_H = \langle 2, h, \bar{h}, H(5, h, \bar{h}, \ell_A, \ell_F, \ell_G) \rangle$

Verification stage for node E**:**

Node E receives ℓ_D, ℓ_F, ℓ_A
Constructs ℓ'_H.
Constructs $\tau_E = H(K^s_e, N, \ell'_H)$

Subscriber gets $\langle \ell_H, \oplus\tau_i \rangle$
Verifies that $\oplus\tau_i$ matches ℓ_H

Figure 5.3: Example aggregation graph for one subscriber S. On the right, we list the labels of the nodes on the aggregation path of publisher E to the subscriber. We then show what labels E receives to construct a verification tag τ_E.

that do not perform aggregation correctly. The verification tags are sent along the aggregation tree and aggregated along the way with the XOR function. Finally, the root node r of the routing tree forwards its label ℓ_r and the XOR of all the verification tags $\oplus_{p_j \in \mathcal{P}} \tau^i_j$ to subscriber s_i. Recall that $\mathcal{P}$ is the set of publishers.

Subscriber s_i can verify that the aggregation of the published data was performed correctly by checking that all the publishers constructed their verification tags using the correct aggregate and commitment values. Subscriber s_i first recomputes each publisher p_j's verification tag τ^i_j using the the root label ℓ_r and shared secret K^i_j. If the XOR of all the computed tags is equal to that received from the root routing node, s_i decides that the aggregate value value is computed correctly.

Figure 5.3 presents an example routing tree with five publisher nodes, and shows how the data published by publisher E is aggregated and verified by the subscriber S.

5.2.2 Verification by Multiple Subscribers

We extend the above setting to allow multiple subscribers to verify an aggregation process. Suppose that there are m subscribers $s_1, \ldots, s_m$ in set $\mathcal{S}$. Let us assume that each publisher p_j shares a secret key K_j^i with each subscriber s_i. At the verification phase, each publisher p_j constructs a separate verification tag τ_j^i for each subscriber S_i, i.e., $\tau_j^i = H(K_j^i, N, \ell_r')$ for $i = 1$ to m. Each routing node aggregates verification tags for each subscriber s_i separately, using the XOR function. Therefore subscriber s_i receives tag $\tau^{s_i} = \bigoplus_{p_j \in \mathcal{P}} \tau_i^{s_i}$.

5.2.3 Analysis

We prove the our basic scheme achieves the sum integrity property and the data privacy property in Section 5.1.2. We only consider the case with a single subscriber node in Section 5.2.1 since, without loss of generality, the arguments in this section hold for the case with multiple subscribers. We first consider the integrity property of the scheme. Our proofs for the integrity property rely on the collision-resistant property of a cryptographic hash function; that is, an adversary cannot construct the same XOR value computed by subscriber s_i with different input values (i.e., a different root label ℓ_r' or different shared secrets) from those used by subscriber s_i.

Lemma 5.2.1 *It is computationally infeasible for an adversary who does not know* *all the shared secrets between the publishers and subscriber s_i to compute the same confirmation tag $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$ that subscriber s_i computes with the shared secrets K_j^i with publisher p_j.*

Proof: Suppose that an adversary does not know a shared secret K_j^i between publisher p_j and subscriber s_i. Since publisher p_j's verification tag τ_j^i has no correlation with the other publishers' tags, the adversary cannot gain any information on XOR of all the verification tags $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$. □

Lemma 5.2.2 *If* XOR *of all the verification tag $\oplus \tau_p^S$ computed by subscriber s_i is same as the* XOR *tag provided by the root routing node, then that root node computes the* XOR *of all the confirmation tags received from the publishers.*

Proof: Since the hash function H used to compute a confirmation tag is collision-resistant, only publisher p who knows a shared secret K_j^i can compute each confirmation tag correctly. Therefore, by Lemma 5.2.1, the only way to computes the correct XOR of the tags $\oplus \tau_p^S$ is to compute the XOR of all the tags provided by the publishers. □

Lemma 5.2.3 *If* XOR *of all the verification tags $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$ computed by subscriber s_i is same as the* XOR *tag provided by the root routing node, then every publisher p_j must have used the same root label ℓ_r to compute its confirmation tag τ_j^i.*

Proof: When subscriber s_i computes the XOR of the verification tags $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$, s_i computes each τ_j^i using the the same root label ℓ_r, which s_i receives from the root routing node. By Lemma 5.2.2, the root routing node obtains the verification tag for subscriber s_i by computing XOR of all the tags from the publishers. Since the hash function H is collision-resistant, each publisher must use the same label ℓ_r and a shared secret K_j^i to compute its verification tag τ_j^i. $\square$

Lemma 5.2.4 *If* XOR *of all the verification tag $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$ computed by subscriber s_i is same as the* XOR *tag provided by the root routing node, then subscriber s_i can be sure that all the publishers compute the same root label value by recomputing the commitment tree.*

Proof: By Lemma 5.2.3, subscriber s_i knows that the verification tag received from the root routing node is computed by taking the XOR of all the publihsers' correct verification tags. This implies that every publisher recomputed the the same commitment tree and verified the same root label. $\square$

Since our basic protocol allows subscriber s_i to confirm that every publisher verifies the commitment tree of the same root label, our protocol provides the same integrity property as that of SHIA [CPS06]. Therefore, we state the following theorem regarding the integrity property of our protocol without proof.

Theorem 5.2.1 *Let A be the final sum received by the subscriber s_i. If the subscriber accepts A, then $A_L \leq A \leq (A_L + \mu r)$ where A_L is the sum of the data values of all the honest publishers, μ is the total number of malicious nodes, and r is the upper bound on the range of allowable values on each node.*

We next show that the basic protocol protects the privacy of each publisher's data from subscriber s_i.

Theorem 5.2.2 *A subscriber s_i cannot learn any information on the publishers' individual values except for the sum A of those values.*

Proof: Subscriber s_i receives only an aggregate sum, the root label ℓ_r, and the XOR of the verification tags $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$. Although both ℓ_r and $\bigoplus_{p_j \in \mathcal{P}} \tau_j^i$ are computed using each publisher's individual values, it is computationally infeasible to obtain those individual values from the output of the hash function H, which provides the preimage resistant property. □

5.3 Collaborative Protocol

In the basic scheme, publishers and routing nodes construct a separate verification tag for each subscriber. Each subscriber is thus able to individually verify the integrity of the aggregation process. In this section, we focus on the problem of collaborative subscribers, with the following properties: (i) subscribers can only verify the integrity of the aggregate together; otherwise they don't learn any information on the validity of the aggregate; and (ii) any subscriber that misbehaves is detected and identified.

We propose a solution where we distribute secret key shares amongst the subscribers so that when the shares are combined with each other, the integrity of the aggregation process can be verified. This is equivalent to a (n, n)-threshold secret sharing scheme.

To demonstrate the difficulty of our problem, we first show an initial attempt and illustrate a critical denial of service attack that a malicious subscriber can launch against the other subscribers. We then present our collaborative solution for preventing this attack.

5.3.1 First Attempt

We modify the basic scheme for multiple subscribers proposed in the previous section by requiring the routing nodes to combine all the subscribers' verification tags with the XOR operation. The combined tag of m subscribers $s_1, \ldots, s_m$ in set $\mathcal{S}$ thus is computed as $\tau = \bigoplus_{s_i \in \mathcal{S}} \tau^i$.

The combined tag τ can then be verified if and only if all the subscribers collaborate to reveal their individual tags as follows. Once each subscriber s_i receives message $\langle \ell, \tau \rangle$ from the routing network, s_i computes its individual verification tag $\tau^i = \bigoplus_{p_j \in \mathcal{P}} H(K_j^i, N, \ell)$ where $\mathcal{P}$ is the set of publishers. Subscriber s_i then broadcasts its individual tag τ^i to the other subscribers. Once a subscriber receives the individual tags of all the other subscribers, it can verify the integrity of the aggregate sum received from the root node of the routing network. The above solution satisfies both the sum integrity and privacy properties. Both properties can be proved using a similar proof to that presented in Section 5.2.3.

The scheme, however, is insecure against malicious subscribers. In particular, a malicious subscriber can force the verification tag to be verified as invalid, *without detection.* To see this, consider a set of m subscribers where subscribers $s_1, \cdots, s_{m-1}$

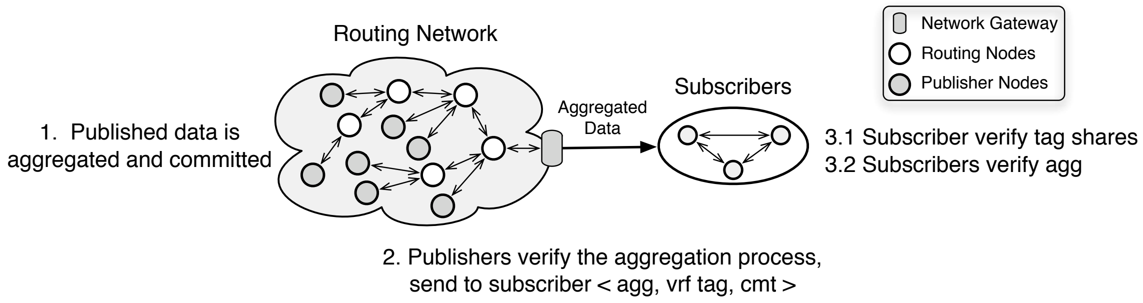

Routing Network
Network Gateway
Routing Nodes
Publisher Nodes
Subscribers
Aggregated Data
1. Published data is aggregated and committed
3.1 Subscriber verify tag shares
3.2 Subscribers verify agg
2. Publishers verify the aggregation process, send to subscriber < agg, vrf tag, cmt >

Figure 5.4: Overview of the PIA collaborative scheme for multiple subscribers.

are honest and broadcast valid tag shares to the other subscribers, and s_m who is malicious, broadcasts an invalid tag share τ^m. Since τ^m is invalid, subscriber s_i where $1 \leq i \leq m-1$ determines that the combined verification tag is invalid. However, it is impossible for s_i to determine if, (i) the tag is invalid because not all publishers agree with the aggregation process, or (ii) it is invalid because a malicious subscriber has broadcast a false verification tag. The honest subscribers are thus denied the service.

5.3.2 Collaborative Scheme

Next we propose a collaborative scheme, which is secure against a privacy-adversary, an integrity-adversary as well as a subscriber-adversary, defined in Section 5.1.3. Intuitively, the scheme builds on our first attempt by first, combining the individual verification tags so that the aggregate can be verified if and only if all subscriber submit their true individual tags to the others. Second, the scheme enables subscribers to verify the correctness of submitted tags from other subscribers. The subscribers are thus forced to behave honestly, or else be detected as malicious.

5.3.2.1 Protocol Overview

The collaborative scheme consists of three stages: (*i*) the aggregation-commit stage, which is identical to that of the basic scheme, (*ii*) a modified verification phase and (*iii*) a new subscriber-verification stage.

In the verification phase, each publisher p_j additionally constructs a commitment to a shared secret K_j^i with a subscriber s_i. Routing nodes aggregate those commitments to shares by jointly constructing a hash tree. In the subscriber-verification phase, subscribers first interact to verify the integrity of the aggregation process by verifying the combined verification tag. If the tag is found to be invalid, each subscriber must then prove the validity of its submitted tag share using its commitment value. We describe our collaborative verification protocol for m subscribers $s_1, \cdots, s_m$ in set $\mathcal{S}$ and n publishers $p_1, \cdots, p_n$ in set $\mathcal{P}$.

5.3.2.2 Setup

Since our protocol requires each subscriber to disclose its secrets at each round of publication, we assume that every pair of subscriber s_i and publisher p_j generates a new shared secret at each round using a pseudo-random number generator (PRNG) with a shared master secret K_j^i as a seed. A PRNG generates a sequence of unpredictable values given a seed; i.e., $PRNG : \mathbb{Z}_l \times \mathbb{N} \rightarrow \mathbb{Z}_l$, where l is the size of a secret. That is, PRNG generates a new secret from a master secret K_j^i and a session sequence number N of each publication. It is computationally infeasible for an adversary to deduce a new secret only from a series of previous secrets without knowing a master secret.

5.3.2.3 Aggregate-Commit-Prove

The aggregate-commit-prove phase is identical to that of the basic scheme, described in Section 4, except for the aggregation of commitments to shared secrets between publishers and subscribers. In particular, assume that publisher p_j has re-constructed the label ℓ of the root node of the aggregation tree. Publisher p_j also computes a verification tag τ_j^i and a commitment c_j^i to a shared secret K_j^i for subscriber s_i, for $i = 1, \ldots, m$:

$$\begin{aligned} \tau_j^i &= H(K_j^i, N, \ell) \\ c_j^i &= H(K_j^i, N) \end{aligned}$$

where K_j^i is the secret shared key between publisher p_j and subscriber s_i and N is a session sequence number. Publisher p_j then forwards $(\tau_j^i, c_j^i),\ i \in [1, m]$ to its parent routing node.

Routing node a_k aggregates, with respect to *each subscriber*, all the commitments and tag values it receives. Suppose a routing node a_k has child nodes $v_1, \cdots, v_t$ and receives message $(\tau_j^i, c_j^i)_{i=1,\cdots,m}$ from child node v_j. Then routing node a_k performs the following aggregations, for $i = 1, \ldots, m$:

$$\begin{aligned} \tau_k^i &= \tau_1^i \oplus \ldots \oplus \tau_t^i \\ c_k^i &= H(c_1^i, \ldots, c_t^i) \end{aligned}$$

Routing node a_k then forwards $(\tau_k^i, c_k^i),\ i \in [1, m]$ to its parent routing node. This continues until the root of the routing tree is reached. The root node of the routing tree, a_r, then computes the combined verification tag using the XOR aggregation operation:

$$\tau = \tau_r^1 \oplus \cdots \oplus \tau_r^m$$

and forwards $\langle \tau, c_r^1, \ldots, c_r^m \rangle$ to the subscribers. For clarity, we present an example of the above process in Figure 5.5. The figure shows how the commitment values of publisher nodes D and E are aggregated along the routing tree.

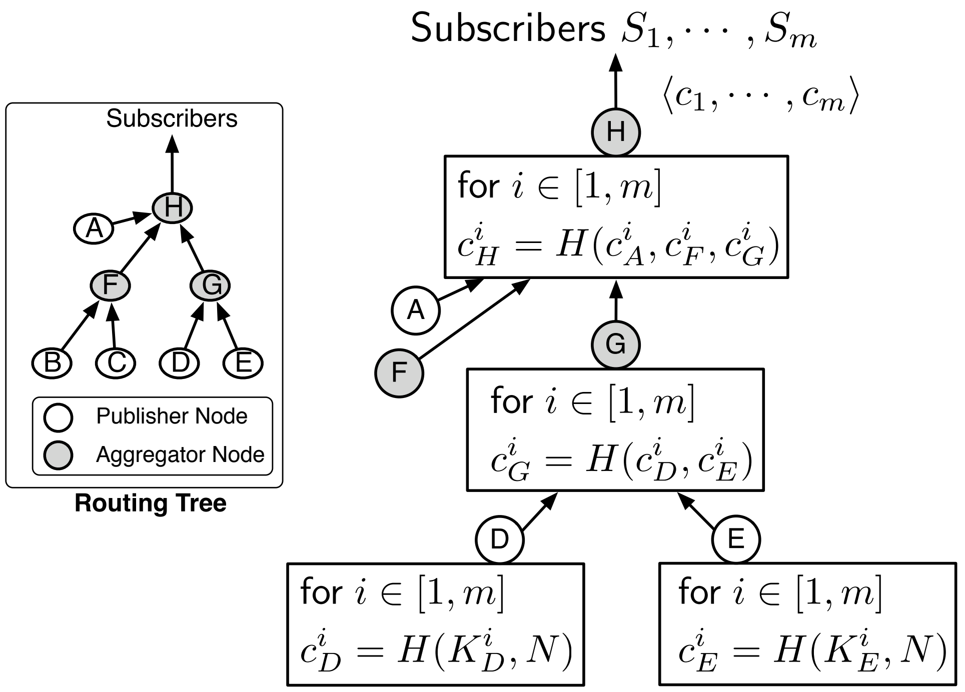


Figure 5.5: Construction of the commitment trees during the aggregate-commit phase.

Note that the hash tree construction we have presented corresponds to what is known as the Merkle hash tree [Mer80, Mer89]. In particular, a Merkle hash tree is used to commit to a set of values, where the committed values correspond to the tree leaf nodes and the value of the internal nodes is derived from the hash of the values of its child vertices.

5.3.2.4 Collaborate-Verification

Each subscriber s_i receives $\langle \ell, \tau, c_r^1, \ldots, c_r^m \rangle$ where ℓ is the label containing the alleged data aggregate and commitment over the aggregation process. Each subscriber s_i does the following:

1. Using ℓ and its shared secrets $K_1^i, \ldots, K_n^i$, construct verification tag share τ^i.

2. Broadcast tag share τ^i to all the other subscribers.

3. Once all tag shares are received, compute combined verification tag $\tau' = \tau^1 \oplus \ldots \oplus \tau^m$ and check if $\tau \overset{?}{=} \tau'$.

4. If $\tau = \tau'$, then we can conclude that the aggregation process has been computed correctly and stop.

   If $\tau \neq \tau'$, then we must determine if aggregation process is incorrect or there are malicious subscribers. Therefore, a prove request is broadcast to all the subscribers.

5. Once a prove request is received, subscribers must prove that they have submitted valid tag shares. To do this, each subscriber s_i reveals enough key information to enable another subscriber s_j to verify that commitment value c^i corresponds to the tag share τ^i. The details of the proof are presented in Section 5.2.3.

   If malicious subscribers are detected and identified, the protocol is stopped.

6. If all subscribers have been verified honest, then we can conclude that the combined aggregation tag is invalid because an aggregating routing node has falsified the aggregation process.

5.3.2.5 Proof Protocol

Subscriber s_i can prove that the submitted verification tag τ^i is related to the commitment value c^i by disclosing the set of secret keys that it shares with all the publisher nodes, i.e., $K_1^i, \ldots, K_n^i$. Then a verifying subscriber s_j can verify that the submitted tag τ^i corresponds to the commitment c^i by:

1. Checking the validity of the submitted keys by computing the commitment $\tilde{c}^i$ using the keys and session sequence number, and checking if $c^i \overset{?}{=} \tilde{c}^i$. (We here assume that each subscriber s_i knows the topology of the routing network to compute the same commitment tree, but we can remove this assumption if the root routing node computes the commitment of all the shared secrets for each subscriber s_i.)

2. If $c^i \neq \tilde{c}^i$, then the submitted keys are invalid and subscriber s_i is identified as malicious. Protocol is stopped.

   If $c^i = \tilde{c}^i$, then the submitted keys are correct.

3. Verifying subscriber s_j then confirms that the verification tag share submitted by s_i is correct by recomputing the share using ℓ, N and the submitted keys. If tag share does not verify, s_i is identified as malicious. Protocol is stopped. Else s_i is honest.

Note that subscribers need to exchange their secrets with each other only when some subscribers fail to verify the verification tag of an aggregate sum. We expect this is a rare case since we expect that a pub-sub system removes malicious subscribers once they are detected.

5.3.3 Analysis

Lemma 5.3.1 *An integrity-adversary does not gain any security advantage by modifying the commitment values.*

Proof: The goal of the integrity-adversary is to force the subscribers to accept a false aggregation value in a stealthy manner. Now consider an integrity-adversary that not only falsified the aggregation operation but also the commitment values. An incorrect commitment value c^i will *only* lead to an inconsistency in the proof protocol submitted by subscriber s_i and the incorrect conclusion that s_i is malicious. Such a conclusion does not add to the security advantage of the integrity-adversary. $\square$

Because of Lemma 5.3.1, henceforth we assume that the integrity-adversary does not falsify the commitment value by deviating from the protocol.

We use Theorems 5.2.1 and 5.2.2 from the basic protocol to show that the aggregation scheme is privacy-preserving and integrity-assured aggregation.

In the following, we assume that all subscribers send messages. A subscriber that stops the protocol by not sending a message always wins and we therefore ignore this case.

Theorem 5.3.1 *Subscriber s_i can verify a combined verification tag τ if and only if it receives the correct verification tag of all the other subscribers, given an alleged aggregate and session sequence number.*

Proof: To verify a combined verification tag, each subscriber s_i independently constructs its tag share τ^i which depends on the secret keys of s_i, the alleged aggregate and the session sequence number. Note that subscriber s_i cannot forge the tag share of another subscriber s_j since it does not have access to its keys (this follows from the security of the cryptographic hash function).

The combined verification tag can only be correctly verified if the computed combined tag is equal to an ideal value that depends on all the subscriber secret keys, the alleged aggregate and the session sequence number. If any of these values are incorrect, then the computed combined tag is not equal to the ideal value and thus the verification fails. $\square$

Theorem 5.3.2 *A malicious subscriber that deviates from the subscriber-verification protocol is detected and identified.*

Proof: A malicious subscriber can either submit an invalid individual verification tag or an invalid set of secret keys during the subscriber-verification protocol. If an invalid set of secret keys is submitted, then the malicious subscriber is always detected and identified because the keys do not match those that generate the commitment tree. This follows from the collision-resistance property of the hash function. If an invalid individual verification tag is submitted, other subscribers can compute the tag with a set of verified secrets and the root label of the aggregation tree provided from the routing network. Because of the collision-resistance of the hash function, a malicious subscriber cannot change the set of secret keys used to generate the commitment. Therefore the malicious subscriber is always detected and identified if it does not use identical sets of keys for the commitment and the verification values. □

Lemma 5.3.2 *The subscriber-verification scheme is forward secure; i.e., even when the subscribers reveal their secret keys, all subsequently executed subscriber-verification protocols are secure.*

Proof: Reduced to the security of the PRNG. □

5.4 Future Directions

In the future, we plan to extending our work in three main directions. First, we would like to improve the efficiency of the subscriber-verification protocol. Currently to verify the honesty of subscribers, the communication cost of each subscriber is $\mathcal{O}(nm)$, where there are n publishers and m subscribers in the network. Although in this work we assume that subscribers are not energy constrained, it would be desirable to reduce the communication exchange between the subscribers. Second, we would like to reduce our trust assumptions, such that the confidentiality of the raw published data need be preserved with respect to routing nodes as well as subscribers. Currently our model assumes that publishers trust the routing nodes with respect to the confidentiality of their raw data. Finally, we plan to extend our secure and verifiable aggregation scheme to support other aggregation functions such as higher order statistics such as quantiles. Currently we can support only sum, mean, standard deviation, count and min/max.

Appendix A

Optimized Bijective Rule

Du and Hwang [DH93] presented a bijective algorithm, where at each step, if an impure set X is discovered, then the algorithm bisects X and tests the resulting two subsets X_1 and X_2. They propose the following partitioning rule:

Algorithm 7 ***Optimized Bijective Rule***

Input: *Set* X*.*

Output: *Result sets* X_1, X_2*, such that* $X_1 \cup X_2 = X$*.*

Bisect X *into subsets* X_1 *and* X_2*, where* X_1 *contains* $2^{\lceil \log |X| \rceil - 1}$ *nodes and* $X_2 = X \setminus X_1$*.*

This rule partitions the network such that the detection tree is made up of one complete and one incomplete binary subtree. Figure 3.1 uses this rule to partition the network of 12 nodes. We refer the reader to the original paper for the proof of the theorem.

Theorem A.0.1 *Given an input set of nodes N, partition degree 2 and the bisecting rule defined by Algorithm 7, identification algorithm 1 produces at most* $2c(\log_2 \frac{|N|}{c} + 1) + 1$ *partitions before outputting all the compromised nodes.*

Figure A.1 shows the behavior of the partition cost of the binary identification algorithm when the Du and Hwang's partition rule is used and compares this cost

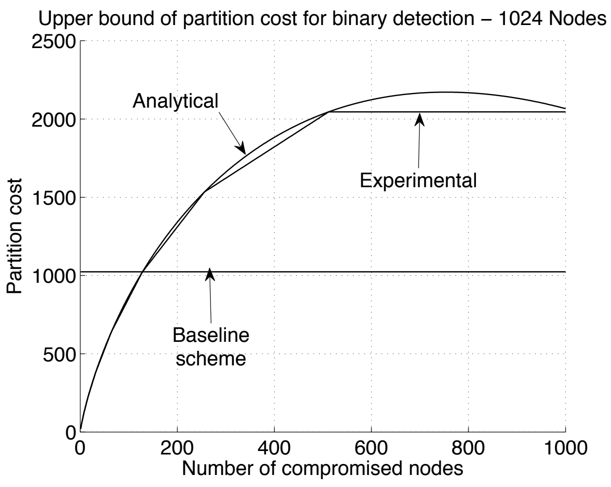


Figure A.1: Comparison of the analytical and experimental bounds with the baseline solution.

with the total cost of the trivial scheme. The figure shows that when considering the partition cost of the identification algorithm alone, the identification algorithm performs better than the trivial solution for up to 128 compromised nodes. Therefore if an adversary compromises more than 128 nodes in the network (or $\frac{1}{8}$th of the total nodes in the network) then aggregation is not a useful primitive for a secure and defensive network.

The figure also compares the analytical partition cost obtained from Theorem A.0.1 and the numerical partition cost computed experimentally. The numerical results yield exact results. Although the analytical curve follows the numerical curve faithfully, it does produce significant error for $c > 128$, where c denotes the number of compromised nodes. The experimental curve reaches a plateau at $c = 128$. This can be explained as if $c = 128$, the worst distribution of compromised nodes requires the testing of every single node; consequently no extra test is required for $c > 128$.

Appendix B

Key Establishment in Heterogeneous Self-Organized Networks [TSN07]

Traditional ad hoc and sensor network settings generally assume a trusted third party (TTP) who is trusted with the keying information and enables secure delivery of keys to the network principals and/or nodes. Security associations, such as authentication of nodes or securing communication channels, are then bootstrapped using this information. In key pre-distribution schemes, the TTP allocates keys to each node prior to deployment either randomly from a key pool [EG02, CPS03], or by using a well-defined combinatorial structure such as a t-design [LS05] that ensures the key subsets allocated to the nodes satisfy certain properties.

However, the assumption of a single TTP can be restrictive in scenarios where the network is self-organized and formed without prior planning. In the following we list some of the immediate applications that require distribution of trust. The first example is in disaster response scenarios where a network may be formed with members belonging to different administrative domains. Furthermore, it might be impossible to access an outside authority due to the lack of preexisting infrastructure or inability to contact off-site systems [LMFJ$^+$04]. In such life-threatening situations, it is not acceptable to deny data from a legitimate principal that might save someone's life. Therefore a 'best-effort' security model might be appropriate in this scenario, allowing strong guarantees when a single TTP can be established and weaker guarantees when no TTP can be assumed. Similarly in combat situations it is essential to allow members of a coalition to join and form collaborative groups. In such dynamic coalitions there is typically no single TTP prior to or during deployment.

Existence of a TTP is also in immediate conflict with privacy enhancing applications. As sensor and ad hoc testbeds have been deployed, it has become clear that user privacy can be easily compromised as a side effect to seemingly innocuous applications [CP03]. For example a humidity sensing network can also be used to monitor activity in a room as the human body effectively alters the room humidity. Therefore by removing the presence of an all knowing authority (i.e., the TTP), communication can be made private to the restricted user set.

Finally, to allow the wide adoption of sensor and ad hoc networks in everyday scenarios, it is desirable to reduce the required knowledge base of network owners. Customers should be able to purchase a set of nodes that are usable upon purchase without requiring the presence of a network administrator. Therefore the node manufacturer can install public data in the nodes that can bootstrap future security associations.

In the following we focus on the problem of group key distribution in self-organized ad hoc and sensor networks where no single point of trust exists. A group key allows nodes to securely communicate with each other and participate in collaborative tasks. The dynamic property of the network allow new nodes to join or exiting nodes to leave the group. This is an essential mechanism in the first two applications listed above. We consider heterogeneous networks consisting of two types of nodes: typical low performance sensor nodes and more powerful nodes with more computation and communication resources. It has been recently shown [DL05, Eco] that networks that consist of homogeneous nodes cannot scale well and also have lower performance compared to networks that include a number of more powerful nodes. Introducing more powerful nodes also improves reliability and lifetime of the network [Eco]. Furthermore [TCC$^+$06] showed that pairwise communication security in the presence of a TTP is not necessarily sacrificed if a key distribution scheme leverages the existence of more capable nodes.

B.0.1 Related Work

The first work on key pre-distribution in ad hoc network without a TTP is due to Chan [Cha04]. In this construction each group member individually selects his keys from a common *public* key pool in a specified way. The aim of the protocol is to probabilistically construct a *Cover Free Family (CFF)* that will ensure shared keys between nodes. After the *key selection phase*, nodes follow a *shared key discovery protocol* that uses homomorphic encryption to discover nodes' shared keys. Chan showed that his scheme allows any two nodes to communicate securely with a high probability and the system provides security against collusion attack. However, [WW05] showed that the probability that the constructed structure is a CFF is very low and so the protocol cannot achieve its stated goal.

The closest work to our scheme is Luo et al. [LSNBS06] who proposes a probabilistic group key management protocol (referred to as LSBS) for ad hoc homogeneous networks. The objective of LSBS is to establish a common shared key for the whole group. The protocol consists of three phases. In the setup phase, each node randomly selects a set of keys from the key pool and performs a shared key discovery (SKD) protocol with each neighboring node to discover shared keys. The group key is generated by special subsets of nodes called initiating groups (IG), and is distributed by flooding the network. Although LSBS protocol achieves its stated goal, in practice there are challenges that if not addressed makes the protocol impractical. In particular, our simulation of LSBS in [TSN07] show the following shortcomings in the protocol.

Firstly, LSBS implicitly assumes that a single IG is formed where in practice many IGs may simultaneously exist. In fact our simulation results show that in a network of 1000 nodes, where each node has a key ring of size 150 keys, we can form up to 100 IGs. To obtain a single group key for all nodes some mechanism for negotiation and/or cooperation among IGs is required, which substantially increases the communication and computation cost which is very undesirable in a resource constrained network. The solutions also needs to be carefully designed to prevent security compromise. The communication cost of the shared key discovery (SSD) phase of the protocol is $\mathcal{O}(l)$ where l is the size of the key ring. LSBS requires a node u to execute the SSD protocol with all of its neighboring nodes. If on average a node is in the neighborhood of d other nodes, a communication cost of $\mathcal{O}(d \cdot l)$ per node is incurred. For networks with battery powered nodes it is essential to reduce this cost in order to prolong network lifetime. Finally, LSBS is analyzed using a simple threat model that does not take into account real life threats in a wide range of application scenarios. The adversary is considered passive and can only eavesdrop on the communications. Given that the key pool is public, the adversary's objective is to either determine the node key or the link key that secures the link between two nodes. In sensor networks it is common to assume that the adversary can compromise a subset of nodes and obtain the secret information of the nodes. Such information includes the key rings of the node and the keys that the nodes share with their neighbors. This latter information will reduce the effort required for finding the key rings of uncompromised nodes, and/or the link keys for links between the compromised node and its neighbor nodes.

B.0.2 Our Contribution

In this appendix, we propose a Layered Key Pre-Distribution (LKD) Scheme for networks of heterogenous nodes: resource constrained nodes and a small number of high performance nodes (level 1) which have more resources and are possibly better protected (e.g., use tamper proof hardware). LKD uses an unbalanced distribution of keys, where high performance nodes are allocated a larger key ring. The level 1-centric clusters that are formed around result in more efficient generation of group keys.

We give a probabilistic analysis of the protocol and show that the inclusion of a small number of more powerful nodes in the network results in constant communication and computation cost, independent of the neighborhood size of a node.

We support our analysis via simulation results. We next evaluate the security of the protocol in a strengthened security model. We argue that with a public key pool and without a TTP, previous proposed threat models and security metrics such as network resiliency [CPS03, EG02], which assumed secret key pool and a TTP, are no longer valid. We update these definitions for our new system and trust model and define a new security metric called neighbor resiliency. We analyze the security of both LKD and LSBS under this new threat model. Our analysis shows that LKD achieves better security than LSBS against node compromising adversaries because sensing nodes in LKD learn much less information about the nodes in their neighborhood.

B.1 System Model

We consider the network to be fully self-organized, meaning that there is no infrastructure (hence no public key (PK) infrastructure). Traditional network models considered for sensor models not only assume a homogeneous network but also assume either a grid or a random graph [EG02, CPS03] model where all neighboring nodes are in communication contact. A more realistic model takes into consideration the various signal-blocking barriers and interference sources such as hills and buildings that exist in the deployed environment. In practice, deployed nodes are often segregated into exclusive neighborhoods due to the features of the landscape [TCC+06]. Our model accounts for this by considering a cluster based network, where sensor nodes form ad hoc groups around more powerful nodes which act as the backbone of the network. Therefore the sensor nodes connect to the rest of the network through the powerful 'gateway' nodes.

We assume a heterogeneous sensor network of size n consisting of two types of nodes: sensing or level 2 (L2) nodes which are resource constrained and have limited storage and energy capabilities and level 1 (L1) nodes which are more capable, with larger memory, more powerful transceivers and energy source. As a result L1 nodes can store larger key rings and other state data as well as communicate with a larger neighborhood of nodes. The network consists of c L1 nodes and $(n - c)$ L2 nodes. Example L2 nodes are small Berkeley Mica2 motes with 8-bit 4MHz processors and 128 KB memories [MIC]. L1 nodes can be more powerful nodes such as laptops or other portable devices. Such devices have better physical protection against compromise, such as the use of tamper resistance hardware. However for simplicity, we assume the same type of protection for L1 and L2 nodes. We also assume that each node u_i has a unique identifier i.

B.1.1 Trust Model

We assume that the network has no central authority or a single TTP. Each node essentially acts as its own domain authority. Public information (e.g. the key pool) is available to all, including malicious parties.

B.1.2 Authentication

Since we do not assume any TTP, it is impossible to establish strong authentication and identification amongst network nodes. We weaken our requirements such that to control the join of malicious nodes to the group, we assume some auxiliary identification mechanism for nodes (e.g., node hardware). Details of such a mechanism is outside the realm of our work.

B.2 Layered Key Pre-Distribution (LKD) Scheme

In this section we describe the LKD scheme to establish both pairwise and group keys in a self-organized network that does not have a TTP. The heterogenous network consists of resource constrained nodes (L2) and more capable nodes (L1) that contain a larger portion of the key pool than L2 nodes. It follows that L1 nodes are able to establish secure links with a larger portion of the nodes. In each neighborhood, local (l, r)-secure groups are established where l denotes the security level and r is the minimum number of nodes in the group. We will show later that r does not effect the security of the protocol and is used for efficiency purposes. Local groups in a neighborhood together generate a cluster group key which are exchanged to contributively generate a network group key. We ensure that the key generated in each layer (i.e. local, cluster or network) is independent. The overall algorithm consists of the following phases: initial setup, neighborhood discovery, cluster and group key generation, join and leave.

In the initial setup phase nodes agree on parameters used in the protocol.

The system parameters include a public key pool and its partition into κ blocks of size m each. The security parameter is l which defines the level of link security by specifying the minimum number of keys two nodes need to share to establish a secure communication channel. The size of the key rings of L1 and L2 nodes are also set to k_A and k_B. We note that these parameters can either be set by the node manufacturers or during an initial setup phase prior to deployment.

A node u_i randomly selects one key from each key block to form a key ring $\{K^i_j\}^k_{j=1}$, where $k = k_A$ or k_B. Let $k_B = \kappa$. Thus an L1 node needs to choose multiple keys from each block. Let $k_A = tk_B + s$, where $t, s \in \mathbb{Z}$. Node selects t keys from block 1 to $(k - s)$ and select $(t + 1)$ keys from blocks $(k - s) + 1$ to block k (in total s key blocks).

B.2.1 Neighborhood Discovery Phase

In this phase, L1 nodes initially send beacons identifying themselves to their neighborhood nodes. The beacon message for L1 node u_i can take the simple syntax of $< i, L1 >$ where i is the node identifier.

An L2 node 'discovers' an L1 node when it hears its beacon message. To establish a secure channel with the L1 and help populate L1's incidence matrix, it runs a secure shared key discovery (SSKD) protocol, reminiscent of [Cha04, LSNBS06]. This SSKD protocol is essentially a privacy preserving set intersection protocol that allows the two participating parties to discover their shared keys from their individual key sets.

For L1 node v_i, the incidence matrix I^i has k columns labeled by the node keys $\{K^i_j\}^k_{j=1}$, and one row for each neighbor. $I^i(j,t) = 1$ if K^i_t is shared with node u_j in the neighborhood of v_i, and zero otherwise. The incidence matrix of v_i can be used to keep an account of the keys shared by the nodes in L1's neighborhood, *given that the keys are shared with* v_i. This property is important as it maintains the optimal privacy for the neighboring L2 nodes. Specifically v_i does not learn any information about the key ring of its neighboring nodes other than the shared key information it learns during the execution of the SSKD protocol.

If an L2 node is not directly connected to an L1 node (i.e. it is isolated from an L2), it simply waits and performs the join protocol after the key establishment protocol is complete. In this step, L1 nodes also discover each other and establish an l-secure channel between pairs of nodes. This communication network forms the backbone of the larger network.

B.2.2 Secure Shared Key Discovery (SSKD)

Consider the case when node u_j wants to discover the keys it shares with node u_i. Let u_i have keys $\{K^i_j\}^l_{i=1}$ and u_j have $\{K^i_j\}^m_{i=1}$, where $l, m \in \mathbb{Z}$. Assume the existence of a homomorphic encryption scheme, where $E_k(m)$ denotes encrypting message m using key k. The SSD protocol is as follows:

1. u_i forms $f_i(x) = \prod^l_{j=1}(x - K^i_j)$ and send to u_j the encrypted coefficients, $E_{K_i}(\cdot)$.

2. u_j computes $z_g = E_{K_i}(rf_i(K^j_g))$ using the homomorphic property of the encryption scheme, where r is a random number. u_j returns z_g to u_i.

3. u_i decrypts z_g to obtain $rf_i(K_g^j)$. If value is zero, then they have a common key.

4. u_i returns to u_j an m-bit bitmap with 1 at bits where $rf_i(K_g^j) = 0$ and 0 elsewhere.

In contrast to LSBS, our SSKD protocol requires the nodes to exchange an m-bit bitmap indicating the shared keys of the participating nodes (step 4). The main reason for this inclusion is that L1 nodes in LKD can select more than one key from each block and so nodes must indicate which key in the block is shared or not, using the bitmap.

B.2.3 Securing Bitmap Transmission

A potential security leakage is the bitmap exchange step of the SSD, which identifies to an eavesdropper the number of keys shared by two nodes. A smart adversary can then compromise a neighboring node which shares the most keys with a target node, as well as reducing the search space for the channel securing key.

The following protocol takes advantage of the privacy preserving characteristics of a homomorphic encryption scheme such as El Gamal [Gam85]. Assume node u_i wants to privately send a k-bit bitmap b to node u_j. We use the *multiplicative homomorphic properties* of the El Gamal [Gam85] encryption scheme for u_i to send b to u_j. Specifically this property is defined as: $E_K(m_1 m_2) = E_K(m_1) \times E_K(m_2)$ where $E_K(m)$ is the encryption of m using key K.

Let the El Gamal public key of u_j be (g, h) and the secret key be $(x = log_g h)$.

$\mathbf{u_j} \rightarrow \mathbf{u_i}$: $r, d \leftarrow \{0,1\}^*$; Send $< C_1, C_2 >=< g^r, h^r \cdot d >$, (g, h)

$\mathbf{u_i} \rightarrow \mathbf{u_j}$: $r' \leftarrow \{0,1\}^*$; Send $< C_3, C_4 >=< C_1 g^{r'}, C_2 h^{r'} \cdot m >$

$\mathbf{u_j}$: bitmap $b = \frac{C_4}{C_3^x \cdot d}$

Node u_j encrypts a dummy message d and sends to u_i the ciphertext and its PK. u_i multiplies the bitmap with the ciphertext and randomizes the message using r'. Using its private key, u_j can decrypt the processed ciphertext and obtain the bitmap. This protocol ensures that the bitmap remains private to u_i, u_j assuming the El Gamal encryption scheme is secure.

By loading nodes with a set of random r values and associated g^r, h^r during the setup phase it is possible to reduce computation to one exponentiation and two multiplications per node. Furthermore we note that although we are using PK cryptography, we do not rely on the existence of a PKI and therefore we preserve the distributed nature of the network. Finally, we point out that this step is only performed once or twice by sensing nodes through out their lifetime. [TSN07] shows further techniques to reduce energy consumption during this step.

B.2.4 Cluster and Group Key Generation

In this phase, L1 nodes v_i use their incidence matrix I^i to assist the nodes in their neighborhoods to initiate local (l, r) groups where a minimum of r nodes share l keys. This is done by finding a set of r rows $\mathcal{R}$ and at least l columns $\mathcal{L}$ in the incidence matrix for which an (l, r)-secure subset can be formed. The formation of the local groups allow v_i to communicate to a group of nodes via multicast thus reducing communication. Also nodes in local groups contribute to the formation of the cluster keys thus preventing the selection of weak keys. Once a local (l, r) group is formed, v_i informs the group members of their group membership using secure channels. Local group members now can communicate securely using their secret group key K_L, which is the XOR of the group shared keys. Each local group L_i in cluster C contributively generate a partial cluster key $K_C^{L_i}$ in order to democratically agree on a cluster key $K_C = \oplus_i \; K_C^{L_i}$.

Potentially two L2 nodes which are not in direct communication can belong to the same local group. In this case, the L1 node can be used as an intermediate routing point to forward messages. Also if L1 nodes use directed antennas, L1 node can group an (l, r) subset together iff they are in the same vicinity.

The group key can be generated similar to the cluster key by requiring nodes to select a key share for the group key along with the cluster key share. L1 nodes then exchange the partial group key generated in their neighborhoods to arrive at the final group key.

B.2.5 Join and Leave

A newly deployed node u_i can join the network by establishing an l-secure channel to a node u_j which already belongs to the secure group. u_j essentially acts for u_i as the 'gateway' to the network. In the case of an L2 node u_i leaving the group, the neighborhood L1 node uses its incidence matrix to determine the effected keys and purging them. Thus the departing node has no information regarding the key rings of the nodes in its neighborhood. Due to space constraint, we refer the reader to [TSN07] for more details of these protocols.

B.3 Correctness Analysis

In this section we show the correctness of the LKD protocol. We say that LKD is *correct* if the protocol allows the 'backbone' L1-network as well as the cluster of L2 nodes around an L1 node, to be connected and thus functioning with a high probability. Later we verify our results by simulation.

In our theoretical analysis we limit the key ring size of L1 nodes $k_A = t \cdot k_B + s$ as follows (k_B is the key ring size of L2 nodes): $t = 1$, $s = [0..k_B]$. We analyze the general case when s can be assigned any value from $[0..k_B]$. We refer the reader to [TSN07] for the special case when $s = k_B$. To establish an l-secure link, two nodes share at least l keys. For readability purposes, in the rest of the paper we use the notation A and B to refer to L1 and L2 nodes respectively.

Let set S consist of the s key blocks from which an L1 node selects two keys and let $\bar{S}$ consist of the remaining $k - s$ key blocks. Let $P_{A,B}(r, l)$ be the probability of r nodes (one L1 node and $(r-1)$ L2 nodes) sharing at least l keys. Let Z_x be the event that r nodes share a key in a given block x. The probability that Z_x occurs, is equal to p_s for blocks $x \in S$, and $p_{\bar{s}}$ for $x \in \bar{S}$. Key collisions for each block can be modeled as independent Bernoulli trials. The generating function for probabilities $P_{A,B}(r, l)$ is calculated as the product of two binomials with success probabilities of

p_s and $p_{\bar{s}}$:

$$f(x) = (p_s x + (1 - p_s))^s (p_{\bar{s}} x + (1 - p_{\bar{s}}))^{k-s} \tag{B.1}$$

Proposition B.3.1 *The probability that the r nodes share exactly l keys is equal to the coefficient C_l of the x^l term in polynomial Equation B.1, and $P_{A,B}(r, l) = \sum_{i=l}^{k_B} C_i$, where C_i is the coefficient of the x^i term in $f(x)$ and k_B denotes the size of the key ring of L2 nodes.*[1]

Proposition B.3.2 *Let $P_{A,A}(2, l)$ be the probability of two L1 nodes sharing at least l keys. Let α, β, γ be non-negative integers satisfying $2\alpha + \beta + \gamma = l$. Let p_i be the probability of sharing i keys for the first s blocks and $\tilde{p}_i$ be the probability of sharing i keys for the remaining $k - s$ blocks. Then:*

$$P_{A,A}(2, l) = \sum_{\substack{\alpha,\beta,\gamma \\ 2\alpha+\beta+\gamma=l}} \tbinom{s}{\alpha} \tbinom{s-\alpha}{\beta}\, p_2^{\alpha}\, p_1^{\beta}\, p_0^{s-\alpha-\beta} + \tbinom{k-s}{\gamma}\, \tilde{p}_1^{\gamma}\, \tilde{p}_0^{k-s-\gamma} \tag{B.2}$$

This proposition is based on the fact that the first s blocks can contribute 0, 1 or 2 shared keys per block, and the last $(k - s)$ blocks can contribute 0 or 1 shared keys per block. In the above formulae, α represents blocks that share 2 keys and β and γ represent blocks that share only 1 key in S and $\bar{S}$ respectively. For a more detailed proof, refer to [TSN07].

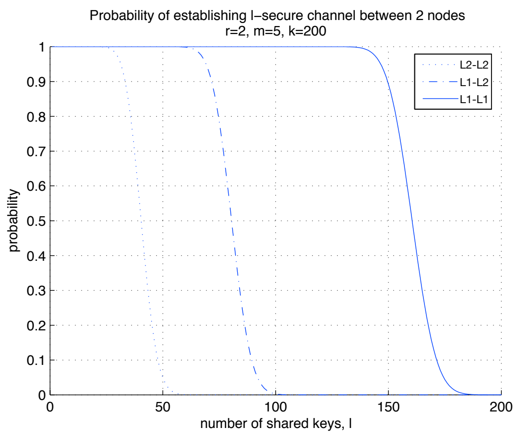
Figure B.1(a) compares the probabilities of two nodes establishing an l-secure channel for different node types, when the key pool is made up of 200 blocks, with a

[1]Examples to illustrate how the above proposition can be used are provided in [TSN07].

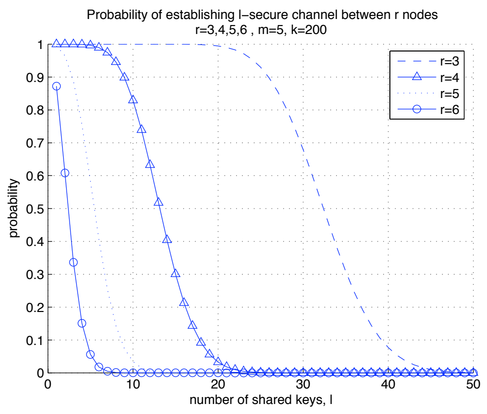
block size of five keys. We can see a rapid transition in the probability of establishing an l-secure channel for different l. Figure B.1(b) generalizes the node pair to groups of r nodes. It is intuitive that establishing an l-secure channel becomes less probable as the group size increases. We also note that when there is a high probability for l-secure channel among r nodes, the probability of establishing a secure channel between two L1 nodes will be an even higher value. It is also interesting to note that the phase transition becomes slower as the number of nodes in the group increases. Figure. B.1(c) graphs the probability of establishing an l-secure channel between an L1 node and an L2 node for different values of s. The results confirm intuition by showing that as the key ring of an L1 node becomes larger, the probability of a secure connection with a L2 node increases. A similar result is verified in Figure B.1(d) when we consider r nodes, consisting of one L1 node and $(r - 1)$ L2 nodes.

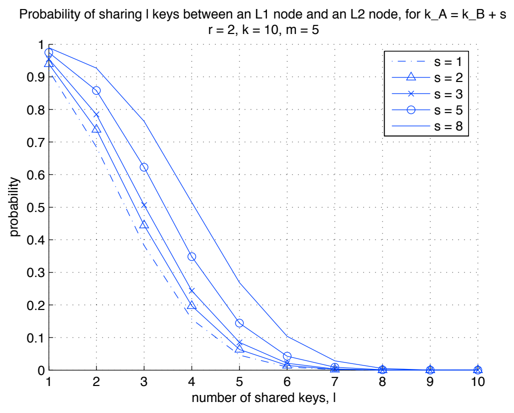
In a more general version of this problem, a node can select extra keys from any block of its choosing, rather than the first s blocks. It is intuitive that in this version of the problem, the probabilities of establishing an l-secure channel do not increase to the same extent as the more special case presented above. We leave the analysis of this problem as a future exercise.

The graphs presented in this section allow a network administrator to choose appropriate values for the system parameters. In the following section, we show how an increased key ring not only increases the probability of establishing a secure channel (as shown), but also decreases the security of the system. It is therefore important to achieve the proper balance between connectivity and security. Section



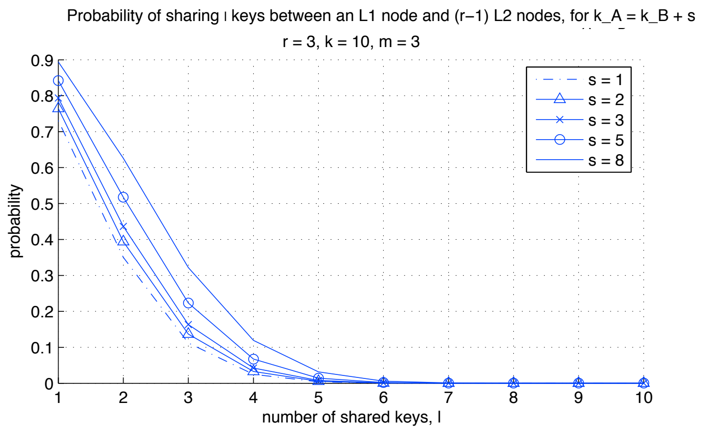
(a) Two nodes, $k_A = 2k_B$





(b) $(r-1)$ L2, one L1 node, $k_A = 2k_B$

(c) Two nodes, $k_A = 2k_B + s$



(d) $(r-1)$ L2, one L1 nodes,$k_A = 2k_B + s$

Figure B.1: Probability of establishing an l-secure channel.

B.5 gives simulation results to confirm the theoretical results.

B.4 Security Model and Analysis

We analyze the security of LKD against two types of adversaries: (i) *Passive* adversary with only access to public data, protocol description and transcript of node communications; (ii) *Node Capturing* (NC) adversary with access to all the information available to a passive adversary, and also the private data of nodes that it has captured. We do not allow a NC adversary to interact with the nodes. That is we only consider the case when the adversary uses its information to eavesdrop on others' communication. The goal of both adversaries therefore, is to learn the secret keys between nodes that are used to secure their links.

The security of traditional key pre-distribution schemes that assume the existence of a TTP [EG02, CPS03, DDH$^+$05, LN03] are based on the facts that (i) the keys in the key pool are exclusively secret to the TTP, (ii) nodes key ring are private, and (iii) the link communication is confidential. In this model an adversary cannot introduce a 'new' device into the network because even if there is no authentication mechanism, it does not have access to the key pool. However by compromising legitimate nodes and obtaining their key rings and/or identities, an adversary can gain entrance into the secure network. The more nodes an adversary compromises, the more it learns of the key pool and the more effective an attack it can launch against a target secure channel. This notion is captured by the *resiliency* of the protocol against node compromise, where resiliency metric is defined to be "the fraction of links in the network a node-compromising adversary is able to eavesdrop on, as a result of recovering keys from captured nodes"[CPS03]. A protocol has stronger security if the adversary is forced to compromise a larger percentage of the nodes to eavesdrop on a target channel. Also, in [EG02, CPS03] information that an NC adversary obtains from captured devices combined with the key indices allows him to gain information about the keys belonging to other network nodes.

The security of the self-organized (SO) protocols do not rest on the secrecy of the key pool; in fact, the key pool is considered to be public information and can be accessed by the adversary. This means that if there are no auxiliary means of authentication, the adversary can introduce a malicious node v with the aim of extracting key information from a victim node u: v can choose a key ring and run SKD with u to find out a subset of keys of u (that they share). It can then select a new key ring and repeat the protocol. After sufficient runs of this, v can learn all the keys of u. This means that it is crucial to assume a method of node authentication that prevents the adversary from introducing nodes of its choice. Since this is not the focus of our paper, we do not consider this scenario and leave it for future work.

The security of the SO protocols is based exclusively on (i) the size of the key pool and (ii) the security of link keys. In LKD, a NC advesary gains only local information from a compromised node; that is, it learns only the key ring of the node and potentially any information it shares with nodes it associates with. In the case of LKD, a node u_i associates only with its neighboring nodes $\mathcal{N}_i$; by compromising u_i an adversary learns not only the key ring of u_i but also the keys it shares with its neighboring nodes. Thus by compromising u_i, the adversary can tighten its search space when attacking (i) a link between two nodes where at least one is neighbor to u_i or (ii) the key ring of a node neighbor to u_i. We capture this notion in the following security parameter for the SO model: *Neighbor resiliency* is defined as the fraction of the key pool the adversary can discard in its exhaustive key search to attack a target secure channel, as a result of recovering keys from neighboring captured nodes. Another security metric we consider is the advantage the adversary gains in determining the key ring of a node when it is in the neighborhood of a compromised node.

In the following, we analyze LKD against first a passive and then a NC adversary. An eavesdropping adversary cannot obtain any information about the keys except to exhaustively guess at the final shared key between nodes. This is because in the course of the key establishment protocol, no information about the key ring of the nodes is leaked. The adversary knows that there are $N = mk$ possible keys and at least l keys from k different possible blocks are used to secure a link. Thus, the search space for the attacker is equal to $\sum_{t=l}^{k} \binom{k}{t} m^l$. Similarly, to determine the key ring of a node of size k, the adversary must exhaustively search $\binom{k}{t} m^l$ possibilities. Due to space constraints, we refer the reader to [TSN07] for the detailed analysis of LSBS.

Because L2 nodes in LKD do not compute an incidence matrix, a compromised L2 node u_c does not leak any keying information about its neighboring nodes. However the adversary does learn (i) The keys that u_c has in common with the L1 node in its cluster, or if it is not connected to an L1 node, the connecting L2 node; (ii) If it is part of an (l, r)-secure local group, only the keys it shares with *all* of them.

Consider three nodes u_i, u_j and u_c. Assume $u_i \in \mathcal{N}_c$, $u_i \in \mathcal{N}_j$, and u_c is a compromised node. Let k be the size of the key rings of u_c, u_i, u_j respectively. The goal of the adversary is to break the secret link between u_i, u_j. By compromising u_c, the adversary obtained the following information: u_c and u_i share b keys and do not share $(k_c - b)$. To guess the key ring of u_i, the adversaries' search space is reduced from m^k to m^{k-b}. The search space to exhaustively guess l shared keys between u_i and u_j is reduced from $\binom{k}{l}m^l$ to $\sum_{\alpha=0}^{l}\binom{k-b}{\alpha}\binom{b}{l-\alpha}m^\alpha$. We can easily see that the search space has been reduced because:

$$\sum_{\alpha=0}^{l}\binom{k-b}{\alpha}\binom{b}{l-\alpha}m^\alpha \leq \sum_{\alpha=0}^{l}\binom{k-b}{\alpha}\binom{b}{l-\alpha}m^l = \binom{k}{l}m^l \tag{B.3}$$

Thus the search space to break an l-secure link between u_i and u_j is equal to:

$$\sum_{t=l}^{k}\sum_{\alpha=0}^{t}\binom{k-b}{\alpha}\binom{b}{t-\alpha}m^\alpha \tag{B.4}$$

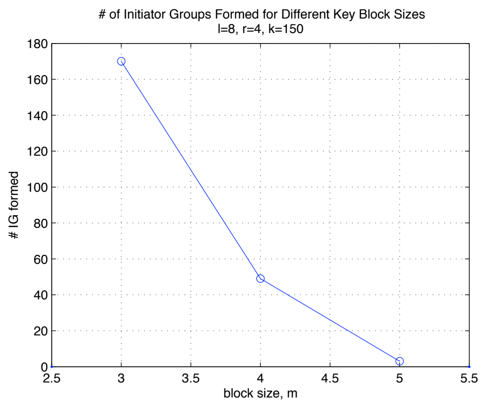
However the number of links and nodes to which these reduced probabilities can be applied to has been decreased dramatically. This is primarily because LKD does not require an L2 node to connect to every node in its neighborhood. Instead the number of secure connections an L2 node needs to establish as well as the keys it shares with neighboring nodes has been reduced to only those that are necessary.

In the event that an adversary compromises an L1 node and the L1 node does not have any tamper resistant hardware, the adversary gains keying information about all the nodes in its neighborhood. In this case the adversary gains as much information as in the LSBS protocol. Since the majority of the nodes in the network are L2 nodes, we can conclude that on average the advantage that an adversary gains by compromising nodes in LKD has been reduced and therefore LKD is more secure than LSBS.

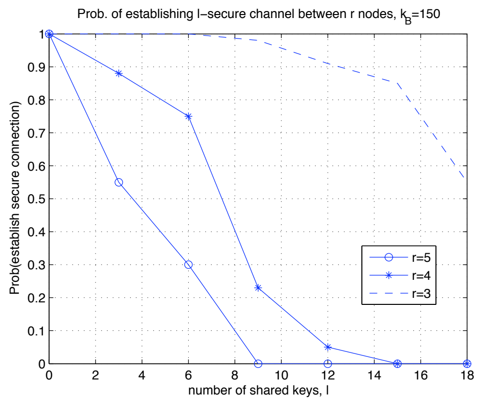
B.5 Simulation and Discussion

The simulation assumes a static network of $n = 1060$ nodes, consisting of 60 L1 nodes and 1000 L2 nodes. This is a reasonable assumption in a dense static network or a highly dynamic network when nodes move around but in a bounded region (e.g a group of rescuers in an emergency situation or troops in a battlefield). We assume that L1 nodes have twice the transmission range R_A of L2 nodes R_B. To guarantee network connectivity and thus allow a large portion of the nodes to participate in the secure group communication, we use the system parameter relationships derived by [EG02] based on the phase transition theory of Erdös and Rényi for connected random graphs. For network connectivity, we require that the neighborhood of each L2 node include 40 other nodes. This is a reasonable assumption used by [EG02, CPS03, TCC$^+$06]. We also need to guarantee that the L1-network (the network of L1 nodes) is connected. Using the area needed for 1000 L2 nodes where the neighborhood of each L2 node has on average 40 nodes, we use 60 L1 nodes where each L1 node is neighbor to 10–15 L1 nodes. At the beginning of the simulation, each node randomly selects a key ring of size $k_A = 300$ for L1 nodes and $k_B = 150$ for L2 nodes. Nodes can establish an l-secure connection by sharing at least l keys.

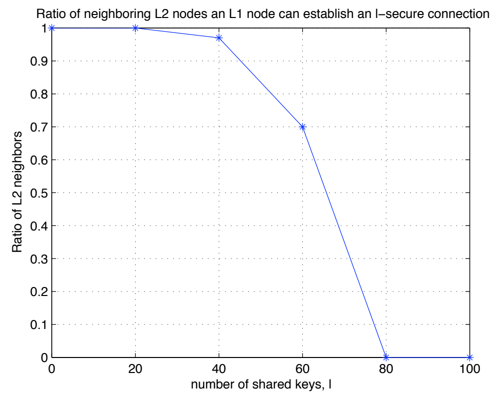
We simulated LSBS using the above configuration in [TSN07] by excluding the L1 nodes. Our results highlighted the shortcomings of LSBS and various practical issues regarding IG formation that were not dealt with in [LSNBS06]. We summarize these as follows: (1) As the number of shared keys needed to establish a secure channel decreases, a larger number of initiator groups get created. This leads to high network communication due to the respective network floods and higher computation load due to the subsequent encryption and decryption of flooded messages; (2) There is a sharp transition rate for the number of IGs formed for different key block sizes m. Fig. B.2(a) shows the jump from very small number of IGs (e.g. $m = 5$) to almost 50 IGs when $m = 4$. However we know that the larger the number of keys shared between two neighbors, the less resilient the protocol is against neighbor-compromise. It is thus important to select network parameters such that allow us to minimize the number of IGs that get created but to also achieve a high degree of security against both an active and a passive adversary. By introducing hierarchy in the LSBS scheme, we are able to better control not only the formation of the local and cluster groups but also the distribution of the group keys. Fig. B.2(b) and (c) show the probabilities of connection for different local group sizes as well how much of the neighborhood can establish a pairwise l-secure connection with an L1 node. Our results show that with very high probability, we can achieve a connected network. In particular, an L2 node can establish a secure connection with an L1 node with very high probability. Fig. B.2(d) graphs the distribution of the size of the (l, r)-groups centering around each L1 node. Each group on average is made



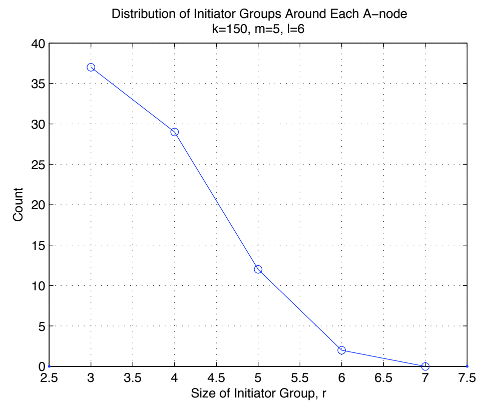
(a) LSBS: #IG created for different key block sizes m



(b) LKD: Prob. that one L1, $(r-1)$ L2 nodes establish l-secure channel.



(c) LKD: Ratio of local L2 nodes that an L1 can establish l-secure channel.



(d) LKD: #groups formed for different security parameters l.

Figure B.2: Analysis of LKD scheme

up of one L1 node and three L2 nodes. We emphasize that the size of a group has no influence on the security of the group key, rather it ensures a more democratic process since more nodes contribute to the calculation of the group key.

Comparing the performance of LKD and LSBS protocols, the necessary resources of a sensing node is reduced in LKD as:

Reduced communication load. The L2-network is no longer flooded with all the partial group keys due to the clustering of the nodes and the management of the local (l, r) groups by the L1 nodes. In particular, each L2 node, with a high probability, needs to only connect to the neighboring L1 node. Furthermore if it falls in an (l, r) group, it needs to exchange $\mathcal{O}(r)$ number of messages to generate a partial cluster and group key. Therefore the number of messages that a sensing node receives and transmits is no longer a function of the neighborhood size.

Reduced computation load. LKD avoids the need for each sensing node to perform multiple decryption and re-encryptions when transporting the group key. In addition the management and decision making required for IG formation has been avoided and made a responsibility of the powerful L1 nodes. In particular in LKD with a high probability, each sensing node performs the SSKD protocol once with the neighboring L1 node. In contrast in LSBS nodes executed the SSKD protocol with every node in their neighborhood (e.g. in our simulation, this would be 40 times).

Reduced storage space. In LKD sensing nodes do not store the incidence matrix which is of the order $\mathcal{O}(k \cdot d)$ where k is the key ring size and d is the size of the neighborhood. Nodes also do not need to keep an account of the different local groups or IGs they belong to.

Finally we note that in LKD, the load on each L1 node is at most equal to the load on *every* node in LSBS. Also, the number of times LKD floods the network of L1 nodes is in the same order as the number of floods of the *whole* network for LSBS.

Bibliography

[AGHS01] David S. Alberts, John J. Garstka, Richard E. Hayes, and David A. Signori. *Understanding Information Age Warfare*. Command and Control Research Program (CCRP) Publications, 2001.

[AGW05] Mithun Acharya, Joao Girao, and Dirk Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 47–53, 2005.

[AK06] W. Ahmad and A. Khokhar. Secure aggregation in large scale overlay networks. In *Proceedings of the 49th Global Telecommunications Conference*, page 2005, 2006.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 563–574, 2004.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neil Koblitz, editor, *Advances in Cryptology - Crypto'96*, pages 1–15, 1996.

[BHGB07] D.E. Bakken, C.H. Hauser, H. Gjermundrod, and A. Bose. Towards more flexible and robust data delivery for monitoring and control of the electric power grid. Technical Report TR-GS-009, Washington State University, May, 2007.

[BKM04] H. Baldus, K. Klabunde, and G. Müsch. Reliable set-up of medical body-sensor networks. In *In Proceeding of Wireless Sensor Networks, First European Workshop (EWSN)*, pages 353–363, 2004.

[Blo70] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[BYKS01] Ziv Bar-Yossef, S. Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *Proceedings of 33rd ACM STOC*, pages 266–175, 2001.

[CCT05] Lifeng Chen, Zhanglong Chen, and Shiliang Tu. A realtime dynamic traffic control system based on wireless sensor network. In *ICPPW '05: Proceedings of the 2005 International Conference on Parallel Processing Workshops*, pages 258–264, 2005.

[CGPM05] Haowen Chan, Virgil D. Gligor, Adrian Perrig, and Gautam Muralidharan. On the distribution and revocation of cryptographic keys in sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):233–247, 2005.

[Cha04] A. Chan. Distributed symmetric key management for mobile ad hoc networks. In *Proc. of Annual Joint Conference of IEEE Computer and Communication Societies, INFOCOM*, volume 4, pages 2414–2424, March 2004.

[CMT05] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, pages 109–117, 2005.

[CP03] H. Chan and A. Perrig. Security and privacy in sensor networks. In *IEEE Computer*, volume 36, pages 103–105, Oct 2003.

[CPS03] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proc. of Symposium on Security and Privacy*, pages 11–14, May 2003.

[CPS06] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 278–287, 2006.

[CRW01] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[DDH$^+$05] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili. A pariwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and Systems Security*, 8(2):228–258, May 2005.

[DDHV03] W. Du, J. Deng, Y. Han, and P.K. Varshney. A witness-based approach for data fusion assurance in WSN. In *Proceedings of the IEEE Global Telecommunications Conference*, 2003.

[DH93] Ding-Zhu Du and Frank K. Hwang. Competitive group testing. *Discrete Applied Mathematics*, 45(3):221–232, 1993.

[DH00] Ding-Zhu Du and Frank K. Hwang. *Combinatorial Group Testing and Its Applications.* On Applied Mathematics, Volume 12. World Scientific, second edition, 2000.

[DL05] X. Du and F. Lin. Improving routing in sensor networks with heterogeneous sensor nodes. In *IEEE Vehicular Technology Conference*, pages 2528–2532, 2005.

[DS05] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In *Proceedings of SPIE Symposium on Smart Structures & Materials / NDE*, 2005.

[Eco] http://www.intel.com/research/exploratory/heterogeneous.htm.

[EG02] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proc. of 9th ACM Conference on Computer and Communications Security*, pages 41–47, Washington, D.C., USA, 2002.

[FD08] Keith Frikken and Joseph A. Dougherty IV. Efficient integrity-preserving scheme for hierarchical sensor aggregation. In *Proceedings of the 1st ACM conference on Wireless Network Security*, 2008.

[Fol08] Mark F. Foley. The dangers of meter data (part 1). *Smart Grid Newsletter*, June 2008.

[FT99] Amos Fiat and Tamir Tassa. Dynamic traitor tracing. In *Advances in Cryptology - CRYPTO'99*, 1999.

[Gam85] T. El Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, July 1985.

[GSW04] J. Girao, M. Schneider, and D. Westhoff. CDA: Concealed data aggregation in WSN. In *ACM Workshop in Wireless Security*, 2004.

[Har89] George W. Hart. Residential energy monitoring and computerized surveillance via utility power flows. *IEEE Technology and Society*, pages 12–16, June 1989.

[HE03] L. Hu and D. Evans. Secure aggregation for wireless networks. In *Workshop on Security and Assurance in Ad Hoc Networks*, pages 384–392, 2003.

[HHSY06] S. Haber, W. Horne, T. Sander, and D. Yao. Privacy-preserving verification of aggregate queries on outsourced databases. Technical Report HPL-2006-128, HP Labs, December 2006.

[HLN$^+$07] W. He, L. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *26th IEEE International Conference on Computer Communications*, pages 2045–2053, May 2007.

[HPP$^+$07] Parisa Haghani, Panos Papadimitratos, Marcin Poturalski, Karl Aberer, and Jean-Pierre Hubaux. Efficient and robust secure aggregation for sensor networks. In *3rd IEEE Workshop on Secure Network Protocols*, pages 1 – 6, Oct 2007.

[KAB$^+$05] Lakshman Krishnamurthy, Robert Adler, Phil Buonadonna, Jasmeet Chhabra, Mick Flanigan, Nandakishore Kushalnagar, Lama Nachman, and Mark Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75. ACM, 2005.

[Khu05] H. Khurana. Scalable security and accounting services for content-based publish/subscribe systems. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 801–807, 2005.

[KNSN05] W. Kastner, G. Neugschwandtner, S. Soucek, and M.H. Newmann. Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203, June 2005.

[LMFJ$^+$04] K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, pages 16–23, Oct 2004.

[LN03] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *Proc. of 1st ACM Workshop on Security of Ad hoc and Sensor Networks*, pages 72–82, Fairfax, Virginia, USA, 2003.

[LS05] J. Lee and D. R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *Proc. of IEEE Wireless Communications and Networking Conference*, volume 2, pages 1200–1205, March 2005.

[LSNBS06] L. Luo, R. Safavi-Naini, J. Baek, and W. Susilo. Self-organized group key management for ad-hoc networks. In *Proc. of ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, 2006.

[MCP$^+$02] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, 2002.

[Mer80] R.C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, 1980.

[Mer89] R.C. Merkle. A certified digital signature. In *Advances in Cryptology - Crypto'89*, pages 218–238, 1989.

[MIC] Mica motes. http://www.xbow.com.

[Mik02] Z. Miklos. Towards an access control mechanism for wide-area publish/subsribe systems. In *Proceedings the 22nd International Conference on Distributed Computing Systems*, pages 516–524, 2002.

[MLWB08] Kazuhiro Minami, Adam J. Lee, Marianne Winslett, and Nikita Borisov. Secure aggregation in a publish-subscribe system. In *ACM Workshop on Privacy in Electronic Society (WPES)*, 2008.

[OP01] L. Opyrchal and A. Prakash. Secure distribution of events in content-based publish subscribe systems. In *USENIX Security Symposium*, 2001.

[PEB06] L.I.W. Pesonen, D.M. Eyers, and J. Bacon. A capability-based access control architecture for multi-domain publish/subscribe systems. In *International Symposium on Applications on Internet*, pages 222–228, 2006.

[PEB07] L.I.W. Pesonen, D.M. Eyers, and J. Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In *International Conference on Distributed Event-based Systems*, pages 104–115, 2007.

[PSP03] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM International Conference on Embedded Networked Sensor Systems*, pages 255–265, 2003.

[RAD78] R. L. Rivest, L. Adelman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–178, 1978.

[RM04] Kay Römer and Friedemann Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, December 2004.

[RPS06] V. Ramasubramanian, R. Peterson, and E.G. Sirer. Corona: A high performance publish-subscribe system for the world wide web. In *Symposium on Networked Systems Design and Implementation*, 2006.

[RR06] C. Raiciu and D.S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Securecomm and Workshops*, pages 1–11, 2006.

[RSPS02] V. Raghunathan, C. Schurgers, Sung Park, and M.B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, Mar 2002.

[SBC+98] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An information flow based approach to message brokering. In *International Symposium on Software Reliability Engineering*, 1998.

[SBD02] Jessica Staddon, Dirk Balfanz, and Glenn Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 122–130, 2002.

[SCD+05] J. A. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood. Wireless sensor networks for in-home healthcare: Potential and challenges. In *High Confidence Medical Device Software and Systems (HCMDSS) Workshop*, 2005.

[SL07] M. Srivatsa and L. Liu. Secure event dissemination in publish-subscribe networks. In *IEEE International Conference on Distributed Computing Systems*, 2007.

[sma] Challenge and opportunity: Charting a new energy future. http://www.energyfuturecoalition.org/pubs/app-smart-grid.pdf.

[TCC+06] P. Traynor, H. Choi, G. Cao, S. Zhu, and T. La Porta. Establishing pairwise keys in heterogeneous sensor networks. In *Proc. of Annual Joint Conference of IEEE Computer and Communication Societies, INFOCOM*, volume 4, pages 2414–2424, March 2006.

[TG08] Gelareh Taban and Virgil D. Gligor. Efficient handling of adversary attacks in aggregation applications. In *13th European Symposium on Research in Computer Security*, 2008.

[TSN07] Gelareh Taban and Reihaneh Safavi-Naini. Key establishment in heterogeneous self-organized networks. In *European Workshop on Security in Ad-hoc and Sensor Networks*, 2007.

[Wag04] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[WCEW02] C. Wang, A. Carzaniga, D. Evans, and A. L. Wolf. Security issues and requirements for Internet-scale publish-subscribe systems. In *Annual Hawaii International Conference on System Sciences*, 2002.

[WGS06] D. Westhoff, J. Girao, and M. Schneider. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution and routing adaptation. *IEEE Transactions on Mobile Computing*, Oct 2006.

[WW05] J. Wu and R. Wei. Comments on "Distributed Symmetric Key Management for Mobile Ad hoc Networks" from INFOCOM'04. Cryptology ePrint Archive, Report 2005/008, 2005. http://eprint.iacr.org/.

[YWZC06] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 356–367, 2006.

[ZDL06] W. Zhang, S. Das, and Y. Liu. A trust based framework for secure data aggregation in wireless sensor networks. In *Proceedings of the IEEE SECON*, 2006.

[ZS06] Y. Zhao and D.C. Sturman. Dynamic access control in a content-based publish/subscribe system with delivery guarantees. In *IEEE International Conference on Distributed Computing Systems*, 2006.