Secure Data Aggregation in Wireless Sensor Networks

Sanjeev SETIA [a],Sankardas ROY [b] and Sushil JAJODIA [b]
[a] *Computer Science Department, George Mason University, Fairfax, VA, USA*
[b] *Center for Secure Information Systems, George Mason University*

Abstract. In many sensor applications, the data collected from individual nodes is aggregated at a base station or host computer. To reduce energy consumption, many systems also perform in-network aggregation of sensor data at intermediate nodes enroute to the base station. Most existing aggregation algorithms and systems do not include any provisions for security, and consequently these systems are vulnerable to a wide variety of attacks. In particular, compromised nodes can be used to inject false data that leads to incorrect aggregates being computed at the base station. We discuss the security vulnerabilities of data aggregation systems, and present a survey of robust and secure aggregation protocols that are resilient to false data injection attacks.

Keywords. Sensor networks, aggregation, security

1. Introduction

Sensor networks are increasingly deployed for applications such as wildlife habitat monitoring, forest fire prevention, and military surveillance [19,21,25]. In these applications, the data collected by sensor nodes from their physical environment needs to be assembled at a host computer or data sink for further analysis. Typically, an aggregate (or summarized) value is computed at the data sink by applying the corresponding aggregate function, e.g., `MAX, COUNT, AVERAGE` or `MEDIAN` to the collected data.

In large sensor networks, computing aggregates *in-network*, i.e., combining partial results at intermediate nodes during message routing, significantly reduces the amount of communication and hence the energy consumed. An approach used by several data acquisition systems [16,31] for sensor networks is to construct a spanning tree rooted at the data sink, and then perform in-network aggregation along the tree. Partial results propagate level-by-level up the tree, with each node awaiting messages from all its children before sending a new partial result to its parent. Researchers have designed several energy-efficient algorithms [16,28] for computing aggregates using the tree-based approach.

Tree-based aggregation approaches, however, are not robust to communication losses which result from node and transmission failures and are relatively common in sensor networks [16,34,35]. Because each communication failure loses an entire subtree of readings, a large fraction of sensor readings are potentially unaccounted for at the data sink, leading to a significant error in the aggregate computed. To address this problem,

researchers have proposed novel algorithms that work in conjunction with multi-path routing for computing aggregates in lossy networks [6,18,20]. In particular, a robust and scalable aggregation framework called *Synopsis Diffusion* has been proposed for computing aggregates such as `COUNT`, `SUM`, `UNIFORM SAMPLE` and `MOST FREQUENT ITEMS` [18,20].

Unfortunately, none of the above algorithms or systems include any provisions for security; as a result, they are vulnerable to many attacks that can be launched by unauthorized or compromised nodes. To prevent *unauthorized nodes* from eavesdropping on or participating in communications between legitimate nodes, we can augment the aggregation and data collection systems with any one of several recently proposed authentication and encryption protocols, e.g., [23,37]. However, securing aggregation systems against attacks launched by *compromised nodes* is a much more challenging problem since standard authentication mechanisms cannot prevent *insider* attacks launched by a compromised node.

Compromised nodes can be used to launch a wide variety of insider attacks that disrupt the operation of the sensor application and network [24,22]. In this chapter, however, we focus on an important class of insider attacks in which the adversary uses compromised nodes to inject malicious data into the network within the framework of a data aggregation system. In particular, we discuss the vulnerabilities of existing data aggregation approaches to such attacks, and present a survey of secure aggregation techniques that are designed to be resilient to such attacks. We also discuss the closely related problem of false data injection in sensor networks in general, and describe an approach that can be used to prevent this attack.

2. Security Challenges

Data acquisition systems for sensor networks can be classified into two broad categories on the basis of the data collection methodology employed for the application:

Query-based systems In query-based systems, the base station (the data sink) broadcasts a query to the network and the nodes respond with the relevant information. Messages from individual nodes are potentially aggregated enroute to the base station. Finally, the base station computes one or more aggregate values based on the messages it has received.

In some applications, queries may be persistent in nature resulting in a continuous stream of data being relayed to the data sink from the nodes in the network. For such applications, the query broadcast by the base station specifies a period (referred to as an epoch); nodes in the network send their readings to the base station after each epoch.

Event-based systems In event-based applications, such as perimeter surveillance and biological hazard detection, nodes send a message to the base station station only when the target event occurs in the area of interest. If multiple reports being relayed correspond to the same event, they can be combined by an intermediate node on the route to the base station.

Data acquisition systems can also be categorized based on how sensor data is aggregated. In single-aggregator approaches, aggregation is performed only at the data sink.

In contrast, hierarchical aggregation approaches make use of in-network aggregation. Hierarchical aggregation schemes can be further classified into tree-based schemes and ring-based schemes on the basis of the topology into which nodes are organized.

As discussed in the introduction, most existing data management and acquisition systems for sensor networks are vulnerable to security attacks launched by malicious parties. Sensor nodes are often deployed in unattended environments, so they are vulnerable to physical tampering. Since current sensor nodes lack hardware support for tamper-resistance, it is relatively easy for an adversary to compromise a node without being detected. The adversary can obtain confidential information (e.g., cryptographic keys) from the compromised sensor and reprogram it with malicious code. Moreover, the attacker can replicate the compromised node and deploy the replicas at various strategic locations in the network [22].

A compromised node can be used to launch a variety of security attacks. These attacks include jamming at the physical or link layer as well as other resource consumption attacks at higher layers of the network software. Compromised nodes can also be used to disrupt routing protocols and topology maintenance protocols that are critical to the operation of the network [14,11]. In this chapter, however, we focus on attacks that target the data acquisition protocol used by the application. Specifically, we discuss attacks in which the compromised nodes send malicious data in response to a query (or send false event reports in event-based systems). By using a few compromised nodes to render suspect the data collected at the sink, an adversary can effectively compromise the integrity and trustworthiness of the entire sensor network.

In event-based systems, compromised nodes can be used to send false event reports to the base station with goal of raising false alarms and depleting the energy resources of the nodes in the network. We refer to this attack as the *false data injection attack*. In Section 3, we discuss an approach for filtering the false data injected by malicious nodes.

Similarly, in query-based systems, compromised nodes can be used to inject false data into the network with the goal of introducing a large error in the aggregate value computed at the data sink. The aggregate computed by the sink is erroneous in the sense that it differs from the true value that would have been computed if there were no false data values included in the computation. Unlike event-based systems, however, in aggregation systems the effectiveness of a false data injection attack depends on both the aggregate being computed, e.g., `MAX` or `MEDIAN`, and whether sensor data is aggregated enroute to the data sink or only at the data sink, and thus the techniques used for preventing this attack differ from the techniques used in event-based systems.

In an aggregation system, a compromised node M can corrupt the aggregate value computed at the sink in four ways. First, M can simply drop aggregation messages that it is supposed to relay towards the sink. This has the effect of omitting a large fraction of sensor readings being aggregated. Second, M can alter a message that it is relaying to the data sink. Third, M can falsify its own sensor reading with the goal of influencing the aggregate value. Fourth, in systems that use in-network aggregation, M can falsify the sub-aggregate which it is supposed to compute based on the messages received from its child nodes.

The first attack in which a compromised node intentionally drops aggregation messages can substantially deviate the final estimate of the aggregate if tree-based aggregation algorithms are used. The deviation will be large if the compromised node is located near the root of the aggregation hierarchy because a large fraction of sensor readings will

be omitted from being aggregated. Countermeasures against this attack include the use of multi-path routing and ring-based topologies [20] as well as the use of probabilistic techniques in the formation of aggregation hierarchies [30].

To prevent the second attack in which a compromised node alters a message being relayed to the sink, it is necessary for each message to include a message authentication code (MAC) generated using a key shared exclusively between the originating node and the sink. This MAC enables the sink to check the integrity of a message, and filter out messages that have been altered. Hence, the effect of altering a message is no different from dropping it, and countermeasures such as multi-path routing are needed to mitigate the effect of this attack.

We refer to the third attack in which a sensor intentionally falsifies its own reading as *the falsified local value attack*. This attack is similar to the behavior of nodes with faulty sensors, and also to the false data injection attack in event-based systems. Potential countermeasures to this attack include approaches used for fault tolerance such as majority voting and reputation-based frameworks [12,15]. In Section 4, we discuss how aggregation schemes can be designed to be resilient to this attack.

The three attacks discussed above apply to both single-aggregator and hierarchical aggregation systems, whereas the fourth attack applies only to hierarchical aggregation systems. This attack in which a node falsifies the aggregate value it is relaying to its parent(s) in the hierarchy is much more difficult to address. We refer to this attack as *the falsified sub-aggregate attack*. In Section 5, we discuss two approaches that have been proposed for resilient hierarchical aggregation that include countermeasures for this attack.

3. Preventing False Data Injection

As discussed in Section 2, in a false data injection attack, an adversary injects false data into the network with the goal of deceiving the base station or depleting the energy resources of the relaying nodes. Several researchers [32,33,36] have proposed solutions to prevent false data injection attacks in sensor networks. Below we briefly discuss the Interleaved Hop-by-Hop Authentication scheme proposed by Zhu *et al* [36] for preventing false data injection attacks.

Interleaved Hop-by-hop Authentication Protocol

We illustrate the false data injection problem and the proposed solution using the scenario depicted in Figure 1 where several sensors been deployed in a cluster for monitoring an area of interest. Events reported by these sensors are relayed to the data collection sink via a multi-hop path formed by several sensor nodes. For ease of exposition, in the description below we will assume that a path between the cluster and the data sink has been established and is stable.

Zhu *et al*'s authentication scheme has two objectives. First, the data sink should be able to verify the authenticity of a report. Second, the nodes on the path from the area of interest to the data sink should be able to filter out false data packets. The scheme is able to meet these objectives as long as the number of compromised nodes is below a threshold number, t.

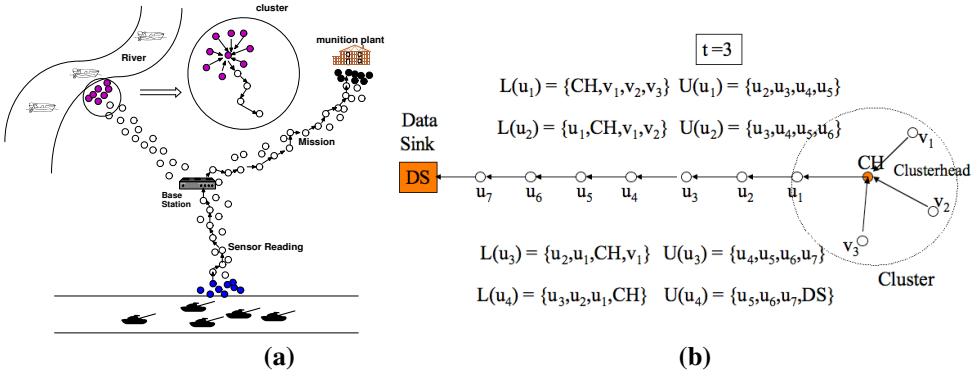

Figure 1. **(a)** A scenario in which sensor nodes have been deployed to monitor three areas of interest – the road, the munitions plant, and the river. **(b)** A logical view of a sensor network in which nodes v_1, v_2, v_3 and CH have been deployed in a cluster to monitor an area of interest. The upper and lower association sets of nodes u_1, u_2, u_3, and u_4 are shown for $t = 3$, where t is the number of compromised nodes that can be tolerated.

Filtering out false data packets within the network (instead of at the sink) has two benefits. First, it results in energy savings since false data packets are immediately dropped instead of being relayed multiple hops to the sink. More importantly, when a false data packet is detected, it reveals the presence of a compromised node. Even though it may be not be possible to identify the compromised node, by detecting the false data close to its origin, it is possible to narrow down the nodes that might be compromised to a small set at a specific location in the network. This information can then be used while taking steps to recover from or mitigate the potential disruptions due to the compromise.

Report Verification To verify the validity of a report, the scheme requires that each report be endorsed by at least $t + 1$ different sensor nodes. To achieve this goal, the sensors monitoring the area of interest are organized into a cluster. If several nodes detect an event of interest, they report the event to the clusterhead, which in turn generates a report for the sink if at least $t + 1$ nodes agree on the event. Note, however, that it is possible that the clusterhead is compromised. Thus, the data sink cannot trust a report simply because it was generated by the clusterhead. An authentication mechanism is needed that allows the data sink to verify that the report was endorsed by $t + 1$ nodes.

Let the event being reported by the nodes in the cluster be E. A straightforward approach to the problem is for each node endorsing the event to compute a Message Authentication Code (MAC) on the event E using its individual key (i.e. key shared exclusively with the data sink) and for the clusterhead to attach $t + 1$ such MACs to the report it sends to the data sink. However, this approach would be very expensive in terms of communication overhead and energy consumption. The solution to this issue is to compress the $t + 1$ individual MACs into a single MAC using the bitwise XOR operation[3]. The report created by the clusterhead then contains the event E, a list of ids of the endorsing nodes, and the compressed MAC. The data sink can thus validate the report by verifying the authenticity of the compressed MAC. (Bellare *et al* have shown this compression scheme is secure [3].)

Enroute Filtering Although this approach will allow the data sink to validate a report, as discussed earlier, it is desirable that a false report be detected as early as possible on the path from the clusterhead to the data sink. To achieve this goal, Zhu *et al* propose to use *interleaved hop-by-hop authentication*.

The basic idea underlying this scheme is that every node en-route to the data sink accepts a report received from an upstream node only if it has been verifiably endorsed by at least $t + 1$ nodes. The scheme requires every node on the path from the cluster generating the report to the data sink to have established security associations (i.e., established pairwise shared keys) with $t + 1$ nodes that are immediately upstream from it. A report is accepted by the node if it has been endorsed by these associated nodes. We refer to the $t + 1$ upstream nodes associated with a node as its lower association set. A node in turn is also in the lower association set of $t + 1$ downstream nodes on the path to the sink. We refer to this set of nodes as the node's upper association set. For nodes that are less than t hops away from the clusterhead, the nodes in its lower association set are designated by the clusterhead from among the nodes in the cluster that reported the event E. (See Figure 1(b)).

To understand how interleaved hop-by-hop authentication works, let us consider a message received by u_4 from u_3 containing a report of an event E. Let $XMAC_{u_4}(u_3, u_2, u_1, CH)$ refer to the compressed MAC created by XORing together the four MACs computed over E using the pairwise keys shared by u_4 with u_3, u_2, u_1 and CH respectively. Then the message received by u_4 should have the following $t+1 = 4$ compressed MACs attached to it – $XMAC_{u_4}(u_3, u_2, u_1, CH)$, $XMAC_{u_5}(u_3, u_2, u_1)$, $XMAC_{u_6}(u_3, u_2)$, and $XMAC_{u_7}(u_3)$. Note that $XMAC_{u_7}(u_3)$ is simply the MAC computed over E using the pairwise key shared between u_7 and u_3, and that $XMAC_{u_4}(u_3, u_2, u_1, CH)$ can be verified by u_4 whereas the remaining three XMACs are destined for nodes u_5, u_6, and u_7 respectively. On receiving a message, if u_4 is able to verify this XMAC, it computes four MACs over E using the pairwise keys it shares with nodes u_5, u_6, u_7, and DS respectively, i.e., the nodes in its upper association set $U(u_4)$. It then XORs these MACs with the XMACs destined for the nodes in its upper association set, i.e., it computes $XMAC_{u_5}(u_4, u_3, u_2, u_1)$, $XMAC_{u_6}(u_4, u_3, u_2)$, $XMAC_{u_7}(u_4, u_3)$, and $XMAC_{DS}(u_4)$. It then attaches these XMACs to the event report E and forwards the message to the node u_5, which will authenticate the message in the same way as u_4 before forwarding it to u_6.

In this manner, at each hop the relaying node can verify if a report is endorsed by $t + 1$ other nodes. As long as the number of compromised nodes does not exceed t, the scheme can detect and filter out false data packets injected into the network within one hop. The size of a message depends upon the number of XMACs attached to a message which is equal to $t+2$, where one of the XMACs is the compressed MAC used by the sink to verify the authenticity of a report and the remaining $t + 1$ XMACs correspond to the XMACs created by the nodes in the receiving node's lower association set as discussed above. Each node has to compute $2(t + 1)$ MACs to verify and authenticate a message. Given that the energy consumed in computing a MAC is roughly equivalent to that used in transmitting one byte [36], this is a beneficial trade.

4. Robust and Secure Single-Aggregator Systems

We now present a survey of secure aggregation techniques proposed for sensor networks. In this section, we discuss approaches for single-aggregator systems in which only the data sink performs the aggregation and there is no in-network aggregation.

In these systems, malicious nodes can attempt to corrupt the aggregate computed at the sink via a falsified local value attack. This attack is similar to the false data injection attack discussed in the previous section, and at first glance it might appear that techniques such as interleaved authentication could also be used to prevent this attack. In event-based systems, however, the nodes monitoring a geographical region simply need to agree that an event of interest has occurred, and thus techniques such as majority voting can be used to filter out malicious reports. In applications that use data aggregation, it is much more difficult and expensive to verify the validity of a sensor reading reported by a node using majority voting. Instead the techniques used to compute the aggregate at the sink need to be designed to be resilient to malicious data.

4.1. Resilient Aggregation Functions

Wagner [29] addressed the following question: in a system with a single aggregator node, which aggregation functions can be securely and meaningfully computed in the presence of a few compromised nodes? Wagner observed that aggregation functions can be viewed as statistical estimators. Based on this observation, he introduced a theoretical framework for modeling the security of data aggregation.

Given a sequence of observations $x_1, \ldots, x_n$ from a known parameterized distribution $p(X \mid \theta)$, where θ is a hidden parameter, the goal of statistical estimation theory is to estimate θ as accurately as possible. Let $X_1, \cdots, X_n$ denote n random variables that are distributed according to a known parameterized distribution $p(X \mid \theta)$, where the hidden parameter θ's distribution is not specified. A parameterized distribution $p(X \mid \theta)$ is a family of distributions, one for each possible value of θ. For instance, $N(\theta, 1)$, the Gaussian distribution with mean θ and variance 1, is a distribution with parameter θ.

An estimator is an algorithm $f : \mathbf{R}^n \rightarrow \mathbf{R}$, where $f(x_1, \ldots, x_n)$ is intended as an estimate of some real-valued function of θ. Let the random variable Θ be equal to $f(X_1, \ldots, X_n)$. Then,

$$\text{root mean square (r.m.s.) error (at } \theta) : \text{rms}(f) = E[(\Theta - \theta)^2 |\, \theta]^{\frac{1}{2}}$$

Note that $\text{rms}(f)$ is a function of θ, the underlying parameter. Also, an unbiased estimator is one for which $E[\Theta |\, \theta] = \theta$ for all θ.

To define a resilient aggregation function, Wagner first defines a k-node attack. A k-node attack A changes k of the input values $X_1, \ldots, X_n$ of an aggregation function. In particular, the attack A is specified by a function $\tau_A : \mathbf{R^n} \rightarrow \mathbf{R^n}$ with the property that the vectors x and $\tau_A(x)$ never differ at more than k positions. We can define the r.m.s. error associated with A by

$$\text{rms}^*(f, A) = E[(\Theta^* - \theta)^2 | \theta]^{\frac{1}{2}} \text{ where } \Theta^* = f(\tau_A(X_1, \ldots, X_n)).$$

In the context of resilient aggregation, Θ^* is a random variable that represents the aggregate calculated at the base station in the presence of the k-node attack A,

and $\mathrm{rms}^*(f, A)$ is a measure of the inaccuracy of the aggregate after the attack. If $\mathrm{rms}^*(f, A) \gg \mathrm{rms}(f)$, then the attack has succeeded in noticeably affecting the estimate of the aggregate. If $\mathrm{rms}^*(f, A) \approx \mathrm{rms}(f)$, the attack has had little or no effect. Wagner defined

$$\mathrm{rms}^*(f, k) = \max \{\mathrm{rms}^*(f, A) : A \text{ is a } k \text{ node attack}\},$$

so that $\mathrm{rms}^*(f, k)$ denotes the r.m.s. error of the most powerful k-node attack possible.

Note that $\mathrm{rms}^*(f, 0) = \mathrm{rms}(f)$. Roughly speaking, we can think of an aggregation function f as being resilient if $\mathrm{rms}^*(f, k)$ grows slowly as a function of k. More formally, an aggregation function f is (k, α)-resilient (with respect to a parameterized distribution $p(X \mid \theta)$) if $\mathrm{rms}^*(f, k) \leq \alpha \cdot \mathrm{rms}(f)$ where α is a real number. The intuition is that for small values of α, a (k, α)-resilient function is a function that can be computed meaningfully and securely in the presence of up to k compromised or malicious nodes.

Wagner also introduces the concept of the breakdown point from robust statistics in the context of resilient aggregation. The breakdown point is defined as

$$\epsilon = \sup\{\frac{k}{n} : \mathrm{rms}^*(f, k) < \infty\}$$

The breakdown point indicates the fraction ϵ of nodes that can be captured before security breaks down. If an ϵ fraction of nodes are compromised, then the r.m.s. error becomes unbounded, and the adversary attains complete control over the aggregation operation. For instance, we can easily verify that Sum has breakdown point $\epsilon = 0$, because any single compromised node can skew the Sum by an arbitrary amount. Consequently, the breakdown point is one measure of the security of an aggregation function against the falsified local value attack. Wagner shows that the breakdown point for aggregates such as `MAXIMUM, MINIMUM` and `AVERAGE` is also 0.

In contrast, `COUNT` can be shown to be a resilient aggregate. Assuming that the inputs X_i follow a Bernoulli distribution with parameter θ (i.e. $P[X_i = 1 |\ \theta] = \theta$ and $P[X_i = 0 |\ \theta] = 1 - \theta$), it can be shown that `COUNT` is (k, α)-resilient for $\alpha = 1 + k.(n\theta(1-\theta))^{-\frac{1}{2}}$. Further, the value of α grows slowly with k when n is sufficiently large.

The aggregate `MEDIAN` is a robust replacement for Average. Since the median represents the sensor reading at the midpoint of the sorted list of all the readings, a 1-node attack can only deviate `MEDIAN` by one place in the sorted list. In general, after a k-node attack, `MEDIAN` will be deviated by at most k places, and when $n > 2k$, `MEDIAN` will be associated with an uncompromised node. Assuming that sensor readings follow a Normal distribution, it can be proved that `MEDIAN` is (k, α)-resilient where $\alpha = (1 + 0.101k^2)^{\frac{1}{2}}$ for large n. Also, the breakdown point of `MEDIAN` is $\frac{1}{2}$.

4.1.1. Filtering Outliers

Wagner also recommended the use of outlier filtering techniques such as truncation and trimming for achieving resilient aggregation. Outliers can be detected and filtered out in a hop-by-hop fashion or at the base station after the the sensor readings reach the base station.

Truncation: One way to make an aggregation function robust to outliers is to place upper and lower bounds on the acceptable range of a sensor reading. If we know that valid sensor readings are usually in the interval $[l, u]$, then we can truncate every input to be within this range. Given any base aggregation function f, we can construct a truncated version by applying f to the truncated data values. The truncated aggregate is an improvement over the conventional aggregate, but it is not an entirely satisfactory solution. A wide interval between the upper and lower bounds gives the attacker a great deal of power, while a narrow interval reduces the utility of the aggregation function.

Trimming: Trimming is a better outlier filtering technique where we ignore the highest ρ and lowest ρ fraction of the sensor readings, and compute the aggregate using the remaining readings, where ρ is a security parameter. Intuitively, we might expect this technique to be fairly robust to outliers, as long as no more than ρ fraction of the sensors are compromised or faulty.

RANBAR: Buttyan *et al* [4] proposed a technique called RANBAR for filtering outliers. RANBAR is based on the use of RANSAC [8], a well-known algorithm used to estimate the parameters of a mathematical model from a set of observed data which contains outliers. Buttyan shows that compared to other resilient aggregation functions such as the trimmed average and the median, RANBAR results in smaller distortion, especially when a large fraction of nodes are compromised.

4.2. SIA: Handling Malicious Aggregators

In the work described above, it is assumed that the aggregator is not compromised. Przydatek *et al* [26] relaxed this assumption in the design of their secure aggregation framework named Secure Information Aggregation (SIA). SIA considers a sensor network where a large number of sensors are deployed in an area distant from a home server (i.e., a user) and a base station is used as an intermediary between the home server and the sensor nodes. After sensor nodes send their readings to the base station, the base station performs the aggregation task and forwards the result to the home server. The base station is the only aggregator node in the network.

The goal of SIA is to enable the user to verify that the answer given by the aggregator in response to a query is a good approximation of the true value even when the aggregator and some fraction of the sensor nodes are corrupted. In particular, SIA includes efficient protocols for secure computation of the `MEDIAN`, `AVERAGE`, `COUNT`, `MINIMUM` and `MAXIMUM` aggregates.

SIA is designed to prevent stealthy attacks, where the attacker's goal is to make the user accept false aggregation results without being detected. In particular, SIA aims to guarantee that if the user accepts an aggregation result reported by the aggregator, then the reported result is close to the true aggregation value with high probability; otherwise, if the reported value is significantly different from the true value due to the misbehavior of the compromised aggregator, the user will detect the attack and reject the reported aggregate with high probability.

The result $\hat{y}$ reported by the aggregator is said to an ϵ-approximation of the true aggregate y if $(1 - \epsilon)y \leq \hat{y} \leq (1 + \epsilon)y$. In addition to the approximation error ϵ, which describes the quality of a reported result, SIA also uses a parameter δ which upper bounds the probability of not detecting a cheating aggregator (i.e., an aggregator reporting a

result not within ϵ bounds). Formally, a protocol is said to an (ϵ, δ)-approximation, if the protocol finds an ϵ-approximation with probability at least $1 - \delta$, and runs in time polynomial in $1/\epsilon$ and $1/(1 - \delta)$.

To achieve its goal, SIA employs an *aggregate-commit-prove* approach. In this approach, the aggregator not only performs the aggregation, but also proves that it has computed the aggregate correctly.

In the first step of this approach, the aggregator collects sensor data from nodes and computes the aggregation result. In the second step, the aggregator constructs a commitment based on Merkle hash-trees corresponding to the sensor data collected in the first step. In this construction, all the collected data is placed at the leaves of the tree, and the aggregator then computes a binary hash tree starting from the leaf nodes; each internal node in the hash tree is computed as the hash value of the concatenation of the two child nodes. The root of the tree is called the commitment of the collected data. Because the hash function in use is collision resistant, once the aggregator commits to the collected values, it cannot change any of the collected values. In the third step, the aggregator sends the the home server both the aggregation result and its associated commitment. The home server and the aggregator engage in an interactive protocol in which the aggregator proves to the home server that the reported results are correct. The interactive protocol used depends upon the aggregate being computed.

To illustrate the aggregate-commit-prove approach, we now discuss how the `MEDIAN` aggregate is securely computed in SIA. First we describe a naive approach for securely computing `MEDIAN`, before discussing the more efficient approach used in SIA.

A straightforward approach for estimating the median using sub-linear communication complexity is random sampling; the user collects a random sample of l measured values via the base station and considers the median of the sample as an approximation of the median of all the measurements. Bar-Yossef *et al* [2] show that $l = \Omega(1/\epsilon^2)$ samples are necessary for an ϵ-approximation of the median.

To reduce the communication overhead, SIA uses an approach based on interactive-proofs. First, the base station sorts the measurements received from the sensors and commits the measurements using a hash-tree construction. Then, the base station engages with the home server in an interactive proof session in which the home server verifies the correctness of the alleged median a_{med} by conducting two tests.

The first test verifies that the committed sequence is indeed sorted. This test can be performed using Sort-Check-II spot checker from [7] with subsequent uniform sampling of pairs of neighboring elements, which requires $O(\log n/\epsilon)$ samples in total. In the second test, the home server checks that a_{med} is sufficiently close to the median of the committed sequence. Here the home server picks $1/\epsilon$ elements from random positions in the committed sequence, and checks that elements picked from the left half of the sequence are smaller than a_{med}, and the elements from the right half are larger than a_{med}.

Przydatek *et al* proved that by requesting only $O(\log n/\epsilon)$ samples from the base station, the home server can check whether the reported value is an ϵ-approximation of the median, and at the same time guarantee a constant probability of detecting a cheating aggregator.

5. Attack-resilient Hierarchical Aggregation

Compared to the single-aggregator scenario, it is far more challenging to design a secure hierarchical aggregation protocol. As discussed in Section 2, the use of in-network aggregation makes it possible for an adversary to launch a falsified sub-aggregate attack. Recently, several researchers have proposed schemes for attack-resilent hierarchical data aggregation [30,27,5]. In this section, we describe two approaches for securing hierarchical aggregation protocols. The first scheme assumes that sensor nodes are organized in a tree-based hierarchy such as that used in the TAG aggregation framework [16], whereas the second scheme assumes the use of a ring-based hierarchy such that used in the Synopsis Diffusion framework [20].

5.1. SDAP

The Secure Hop-by-hop Data Aggregation Protocol (SDAP) [30] is based on the principles of divide-and-conquer and commit-and-attest. We first present a high level overview of the protocol.

SDAP uses a novel probabilistic grouping technique to dynamically partition the nodes in a tree topology into multiple logical groups (subtrees) of similar sizes. An aggregate is computed within each logical group using a standard hop-by-hop aggregation protocol. The leader of each logical group transmits the group's aggregate to the base station, along with a commitment that is generated based on the contributions of each node in the group. After the base station has collected all the group aggregates, it uses an outlier detection algorithm to identify groups whose contribution is potentially suspect. Finally, each group under suspicion participates in an attestation process to prove the correctness of its group aggregate. The base station discards any suspicious aggregates from groups that fail the attestation procedure. The final aggregate is calculated over all the group aggregates that have either passed the outlier detection algorithm or the attestation procedure.

SDAP assumes that the base station cannot be compromised and that a secure broadcast authentication mechanism (e.g., μTESLA [23]) is available. It also assumes that every sensor node has an individual secret key shared with the base station. Further, there is a unique pairwise key shared between each pair of neighboring nodes.

We now discuss the four main phases of SDAP – query dissemination, probabilistic grouping and data aggregation, suspicious group detection, and group attestation – in more detail.

Query Dissemination: In the query dissemination phase, the base station broadcasts the aggregation query message throughout the network. The aggregation tree is also constructed in this phase, if it does not already exist. μTESLA is used for authenticating the broadcast message.

Probabilistic Grouping and Data Aggregation: In this phase, SDAP randomly groups all the nodes into multiple logical groups and performs aggregation within each group. Probabilistic grouping is achieved via the selection of the leader node for each group. Because grouping is a dynamic process, a node will not know in advance whether it will become a group leader or which group it will belong to.

Group leaders are selected dynamically based on the number of nodes in its sub-tree that have not yet been grouped (referred to as the node's *count value*) and a grouping seed

S_g. While the grouping seed is included in the broadcast query, each node calculates its count value based on the count values received from its children during the aggregation process (as discussed below).

Two functions are used in group leader selection. One is a cryptographically secure pseudo-random function H that uniformly maps the inputs (the node id and S_g) into the range of $[0, 1)$; the other is a grouping function F_g that takes the local node's count value as the input and outputs a real number between $[0, 1]$. More specifically, each node, say x, decides if it is a leader node by checking whether

$$H(S_g|x) < F_g(c)$$

where c is the count value of node x. If this inequality is true, node x becomes a leader. Once a node becomes the leader, all the nodes in its subtree that have not been grouped yet become members of its group. The function $F_g()$ is constructed such that $F_g(c)$ increases with the count value c. This ensures that nodes with more descendants have a higher probability of becoming group leaders.

In SDAP, a node's aggregation message contains its node id, its count value, and its aggregate. In addition, each message includes a flag that indicates whether the aggregate needs to be aggregated further by the nodes enroute to the base station. The pairwise key shared between each pair of parent and child is used to encrypt the aggregate. Further, each message includes a message authentication code (MAC) that is computed using the nodes individual key (shared with the base station).

Data aggregation starts from the leaf nodes. Since a leaf node u does not do any aggregation, it just sends a packet containing its id, aggregation flag set to '0', its data, and its count value as '1' to the parent, say v. The above information is encrypted using the pairwise key K_{uv}. The message also includes a MAC generated using node u's individual key K_u that authenticates the information in the message.

When a non-leaf node v receives an aggregate from a child node, it first checks the aggregation flag. If the flag is '0', it performs further aggregation; otherwise, the node directly forwards the packet to its parent node. A non-leaf node aggregates its own reading with all the aggregates received from child nodes with aggregation flag '0'. A new count is calculated as the sum of the count values in the received aggregates with flag '0' plus one. The node v checks if it is a group leader using its own id and the new count as the inputs to the functions $F_g()$ and H. The node v then encrypts the new count value and aggregation data using the pairwise key shared with its own parent.

Node v's message also includes a MAC which is calculated over not only the above fields but also the XOR of the MACs received from its children. In this way, the MAC computed by a node represents the authentication information of all the nodes contributing to its aggregate.

Now suppose that a node x has processed the aggregates from its child nodes and it finds out that it is a group leader. Like any other node, it also computes a new aggregate and appends a corresponding MAC calculated using its individual key shared with the base station. Unlike other (non-leader) nodes, it encrypts the new aggregate with its individual key and sets the flag to '1' in its aggregation packet, so that data from this group will not be aggregated any more enroute to the base station. Note that the MAC computed by a group leader represents an aggregation commitment, which is used in the attestation phase of the protocol.

Determining Suspicious Groups: A compromised node can be expected to forge its sub-aggregate so that it will have a non-trivial influence on the final result; otherwise the attack would not gain much. Thus, the aggregate of a group which contains a compromised node can be treated as an outlier compared to the aggregates of other groups. SDAP uses Grubbs' test [10], also known as the maximum normalized residual test, to detect outliers. The original Grubb's test cannot be directly applied for this purpose because it can handle only univariate data where a group's aggregate has two dimensions, the count value and the aggregate being computed. Yang *et al* modified Grubb's test so that it can detect outliers in bivariate data. Groups whose aggregate values are outliers are considered suspicious, and their aggregates are verified using the attestation process described below.

Group Attestation: The BS broadcasts an attestation message including the id of the leader for the group to be attested, a random number S_a, and S_g. S_a is used as the seed for the attestation and it will determine a unique and verifiable attestation path whereas S_g is included for identifying the query.

During the attestation process, a physical attestation path between the group leader and a leaf node in its logical subtree is dynamically formed. After the group leader node receives the attestation request from the base station, it selects the next hop on the attestation path using a probabilistic procedure in which the probability of a child node to be selected on the path is proportional to its count value reported in the aggregation phase. A selected child runs the same process to select one of its own children to be on the attestation path. Recursively, an attestation path between the leader and a leaf node in the logical group subtree is formed.

Subsequently, each node on the path sends back to the base station its count value and its own reading. If the node is not a leaf node, its parent also asks its sibling nodes to send back their count values, sum values, and their MACs.

After the base station decrypts the received data, it first verifies whether the responding nodes are really the nodes on the attestation path based on S_a, the nodes' ids, and the counts. It verifies whether the count value of every node is the sum of its children's counts plus one. If this check succeeds, it aggregates the data and reconstructs the aggregation result of the group to examine whether nodes on the path have forged the aggregation results in the aggregation phase.

The commit-and-attest technique used by SDAP ensures that once a group has committed its aggregation result, every node involved in the attestation phase has to report its original aggregate. Otherwise, the base station will detect the attack by finding the inconsistency between the committed aggregate and the reconstructed aggregate. Due to the probabilistic grouping scheme, an attacker cannot selectively compromise nodes – for example, making no more than one compromised node appear in the same group. Also, because the aggregates from all the groups are encrypted, a compromised node cannot know if its own aggregate will become an outlier after Grubbs' test is run. Further, a node cannot know whether it will be selected on the attestation path because the attestation path is also dynamically selected in a probabilistic fashion.

5.2. Attack-resilient Synopsis Diffusion

Tree-based aggregation approaches are vulnerable to communication losses which result from node and transmission failures and are relatively common in sensor networks [16,

34,35]. Because each communication failure loses an entire subtree of readings, a large fraction of sensor readings are potentially not incorporated in the final aggregate at the querying node. Although protocols such as SDAP and the resilient aggregation scheme recently proposed by Chan *et al* [5] make the tree-based aggregation approach resilient to malicious data, they remain vulnerable to communication loss.

To address this problem, researchers have proposed the use of multi-path routing techniques for forwarding sub-aggregates [16]. For aggregates such as `MIN` and `MAX` which are duplicate-insensitive, this approach provides a loss-tolerant solution. For duplicate-sensitive aggregates such as `COUNT` and `SUM`, however, multi-path routing leads to double-counting of sensor readings, resulting in an incorrect aggregate being computed. Researchers [6,18,20] have presented novel algorithms that solve the double-counting problem associated with multi-path approaches. A robust and scalable aggregation framework called *Synopsis Diffusion* has been proposed for computing aggregates such as `COUNT, SUM, UNIFORM SAMPLE` and `MOST FREQUENT ITEMS`.

Synopsis Diffusion, however, does not include any provisions for security, and a compromised node can launch several attacks against this framework which can potentially cause the querier to accept an incorrect result. Roy *et al* [27] propose attack-resilient aggregation protocols within the synopsis diffusion framework to compute aggregates such as `COUNT` and `SUM`. These secure protocols are developed by augmenting the original synopsis diffusion algorithms with authentication techniques.

Before discussing Roy *et al*'s protocol, we provide an overview of synopsis diffusion, and discuss how a compromised node could launch a falsified sub-aggregate attack against the protocol.

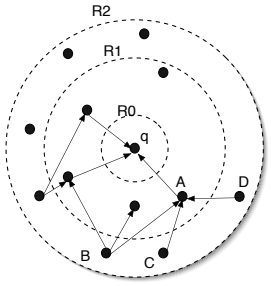

Figure 2. Synopsis Diffusion over a rings topology

Synopsis Diffusion There are two primary elements of the synopsis diffusion approach – the use of a ring-based topology instead of a tree-based topology, and the use of duplicate-insensitive algorithms for computing aggregates based on Flajolet and Martin's algorithm for counting distinct elements in a multi-set [9]. Figure 2 illustrates how this approach uses a rings topology for aggregation. In the query distribution phase, nodes form a set of rings around the querying node q based on their distance in hops from q. During the subsequent query aggregation period, starting in the outermost ring each node generates a local synopsis $s = SG(v)$ where v is the sensor reading relevant to the query, and broadcasts it. (Here $SG()$ is a order- and duplicate-insensitive aggregate function.) A node in ring R_i will receive broadcasts from all the nodes in its range in ring R_{i+1}.

It will then combine its own local synopsis with the synopses received from its children using a synopsis fusion function $SF()$, and then broadcast the updated synopsis. Thus, the fused synopses propagate level-by-level until they reach the querying node, who first combines the received synopses with its local synopsis using $SF()$ and then uses the synopsis evaluation function $SE()$ to translate the final synopsis to the answer to the query.

The functions $SG()$, $SF()$, and $SE()$ depend upon the target aggregation function. We now describe synopsis diffusion algorithms for the `COUNT` aggregate. In the `COUNT` algorithm, each node generates a local synopsis which is a bit vector ls of length $k > \log n$, where n is an upper bound on the nodes in the network. To generate its local synopsis, each node executes the function $CT(X, k)$ where X is the node's identifier and k is the length of ls in bits. $CT()$ can be interpreted as a coin-tossing experiment (with the random binary hash function $h()$ simulating a fair coin-toss), which returns the number of coin tosses until the first heads occurs or $k + 1$ if k tosses have occurred with no heads occurring. In the local synopsis ls of node X, a single bit i is set to 1, where i is the output of $CT(X, k)$. Thus ls is a bitmap of the form $0^{i-1}1\cdots$ with probability 2^{-i}.

Algorithm 1 $CT(X, k)$

```
i=1;
while i < k + 1 AND h(X, i) = 0 do
   i = i + 1;
end while
return i;
```

For `COUNT`, the synopsis fusion function $SF()$ is simply the bitwise Boolean OR of the synopses being combined. Each node fuses its local synopsis ls with the synopses it receives from its children by computing the bit-wise OR of all the synopses. Let S denote the final synopsis computed by the querying node by combining all the synopses received from its children and its local synopsis. We observe that S will be a bitmap of length k of the form $1^{r-1}0\cdots$. The querying node can estimate `COUNT` from S via the synopsis evaluation function $SE()$: if r is the lowest-order bit in S that is 0, the count of nodes in the network is $2^{r-1}/0.7735$. Intuitively, the number of sensor nodes is proportional to 2^{r-1} since no node has set the rth bit while computing $CT(X, k)$.

Attacks Just as in the case of a tree-based aggregation protocol, a compromised node can cause the synopsis diffusion algorithms to output an incorrect aggregate by manipulating a sub-aggregate or its own local sensor reading. We now discuss this attack in the context of the `COUNT` aggregate.

Since the base station estimates the aggregate based on the lowest-order bit r that is 0 in the final synopsis, a compromised node would need to falsify its own fused synopsis such that it would affect the value of r. It can accomplish this quite easily by simply inserting ones in one or more bits in positions j, where $r \leq j \leq k$, in its own fused synopsis which it broadcasts to its parents. Let r' be the lowest-order bit that is 0 in the corrupted synopsis, whereas r is the lowest-order bit that is 0 in the correct synopsis. Then the base station's estimate of the aggregate will be larger than the correct estimate by a factor of $2^{r'-r}$. We also observe that even a single node can launch this attack with

a high rate of success because the use of multi-path routing makes it highly likely that the falsified synopsis will be propagated to the base station.

On the other hand, it is very hard to launch an attack which results in the aggregate estimated at the sink being lower than the true estimate. This is because setting a bit in the falsified synopsis to 0 has no effect if there is another node X that contributes a 1 to the same position in the fused synopsis. To make this attack a success the attacker has to compromise all the possible paths from node X to the sink so that X's 1 cannot reach the sink, which is hard to achieve.

In the rest of this chapter, we restrict our discussion to the previous attack where the goal of the attacker is only to increase the estimate. We note that Garofalakis *et al* [13] have proposed defenses against attacks aimed at decreasing the aggregate estimated at the sink. However, their work does not focus on networks with resource-constrained nodes. Their approach assumes the use of digital signatures and does not consider the use of multi-path routing by the aggregation system.

ARSD Roy et al [27] propose attack-resilient aggregation protocols within the synopsis diffusion framework to compute aggregates such as `COUNT` and `SUM`. These protocols enable the base station to detect the attack and to filter out the false data injected by the attacker so that the estimate of the aggregate is not affected.

In the Attack-Resilient Synopsis Diffusion (ARSD) protocol, a subset of the nodes responding to a query include along with their synopses a message authentication code (MAC) that can be used by the base station to verify the validity of their contribution to the aggregate function. The key observations behind the design of ARSD are that:

- In order to derive the correct estimate of the aggregate (`COUNT` or `SUM`) from the final synopsis (say S) computed at the base station, it is only necessary to figure out the correct lowest order bit (say r) in S that is 0.
- The number of nodes contributing a 1 to bit j decreases as we move from the lowest order bit ($j = 1$) to higher order bits of the synopsis. For example, in the case of `COUNT`, on average, half the nodes in the network will contribute to the leftmost bit of the synopsis, one-fourth of the nodes contribute to the second bit of the synopsis, and so on.

Thus, it is expected that only a small number of nodes will contribute to the bits around the expected value of r; the expected number of nodes that contribute to bits i, where $r < i \leq k$ in the synopsis (k is the length of the synopsis) is very small. In fact, it can be showed that expected number of nodes contributing to the bits to the right of bit r is less than $1/\phi \approx 1.29$. In this approach, nodes contributing to bit i, $i \geq r - 2$ include along with its response to an aggregation query a MAC computed using a pairwise key shared exclusively with the base station. These MACs enable the base station to verify the computed aggregate and to filter out the contributions of the falsified sub-aggregates injected by the compromised nodes.

To design the attack-resilient `COUNT` protocol, two issues need to be addressed. First, since a node cannot locally determine the position of bit r in the final synopsis, the base station needs to specify a global criterion which determines if a node needs to include a MAC along with its synopsis. Second, this criterion should be designed so that the number of such nodes who include a MAC is minimized.

Roy *et al* [27] present two approaches to address these issues. In the basic approach, it is assumed that the base station has an estimate of the lower bound and the upper

bound of Count which are denoted by l_c and u_c respectively. Based upon these bounds, the base station knows that bit r will lie between a and b, which are the bit positions in the synopsis S corresponding to l_c and u_c respectively, i.e., $a = \lceil \log_2(\phi l_c) \rceil$ and $b = \lfloor \log_2(\phi u_c) \rfloor$. Thus, there is no need for the base station to verify the bits to the left of a; only nodes contributing to bits in the range a to b need to prove to the base station that their contribution to the synopsis S is valid.

The collection of bits in the range a to b in synopsis S is referred to as the *synopsis-edge*. It is easy to see that the length of the synopsis-edge is $(\lfloor log_2(\frac{u_c}{l_c}) \rfloor + 1)$ bits. If we denote the number of nodes contributing to the synopsis-edge by η, then, by the properties of the COUNT synopsis, $\eta \leq (\frac{u_c}{2^a} + \ldots + \frac{u_c}{2^b}) \approx \frac{1}{\phi} \cdot (\frac{2u_c}{l_c} - 1)$.

Thus, in the basic protocol, each node checks whether it contributes to the synopsis-edge, and if so, it generates a MAC and forwards the MAC along with its fused synopsis. Specifically, if node X contributes to bit i in the synopsis-edge, it generates a MAC, $M = MAC(K_X, m)$ over the message m whose format is $[X|i]$. After the base station receives the MAC, it checks its validity by executing the synopsis generation function $SG()$ with the inputs for node X. If the verification fails, the contribution of X is discarded. The bits in the synopsis-edge for which the base station has not received a valid MAC are reset to 0. The bits at positions to the left of the synopsis-edge are set to 1. Finally, the base station computes the COUNT aggregate using the synopsis evaluation function $SE()$.

Although a compromised node can falsely set some bits in its local fused synopsis, the base station will be able to discard them by verifying the corresponding MACs. This implies that the attacker cannot falsely increase the COUNT. On the other hand, the attacker may attempt to decrease the estimated COUNT by dropping a genuine MAC (or by corrupting a genuine MAC) sent by a contributing node, but the genuine MAC is likely to reach base station via an alternate path. If base station receives at least one valid MAC for each 1 bit in the synopsis-edge, then base station obtains the true estimate of COUNT.

The basic protocol outlined is not scalable if the ratio of the upper bound (u_c) of COUNT to the lower bound (l_c) is high, because in the worst case a node has to forward $\frac{1}{\phi} \cdot (\frac{2u_c}{l_c} - 1)$ MACs. To address this problem, Roy *et al* propose another protocol which results in lower communication overhead at the expense of greater latency.

In the extended protocol, the synopsis is computed by a sliding window approach where the aggregation phase is divided into multiple epochs. Starting at the rightmost bit k, the extended protocol proceed from right to left within the synopsis S using a bit window of w bits. In each epoch, only the nodes that contribute a 1 to the bits in S corresponding to the current position of the window, send MACs to the base station that can be used to verify their contribution to S. In other words, in the first epoch, only nodes that contribute a 1 to bits k to $k - w + 1$ respond to the query. In epoch two, nodes that contribute to bits between $k - w$ and $k - 2w + 1$ respond, and so on.

The algorithm terminates when the base station has determined that the remaining bits of S to the left of the current window are likely to be 1 with high probability. Once the base station determines that the algorithm can terminate it broadcasts a STOP message to the network to announce the end of the iterative aggregation procedure.

Roy *et al* show via analysis and simulation that the iterative procedure outlined above has low communication overhead in comparison to the basic protocol. They also show that this algorithm results in the same estimate as the original synopsis diffusion

approach when there are no compromised nodes, while providing additional resilience in the presence of compromised nodes.

6. Conclusions

In this chapter, we have discussed the security vulnerabilities of data aggregation protocols for sensor networks. We also presented a survey of secure and resilient aggregation protocols for both single-aggregator and hierarchical systems.

A number of challenges remain in the area of secure aggregation for sensor networks. Secure tree-based aggregation protocols [30,5] remain vulnerable to message losses either due to node failure or compromised nodes. The performance and security tradeoffs between resilient tree-based approaches and multi-path approaches such as Attack Resilient Synopsis Diffusion [27] have yet to be explored. The research community is yet to design a secure aggregation protocol for computing holistic aggregates such as Order-statistics and Most Frequent Items. Finally, many data acquisition systems use persistent queries in which nodes periodically send readings to the sink resulting in streams or flows of sensor data [17,1]. These systems make extensive use of data aggregation. Issues in securing such sensor data streaming applications remain to be investigated.

References

[1] Daniel J. Abadi, Wolfgang Lindner, Samuel Madden, and Jörg Schuler. An integration framework for sensor networks and data stream management systems. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 1361–1364. Morgan Kaufmann, 2004.

[2] Ziv Bar-Yossef, S. Ravi Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. *Proc. of 33rd STOC*, pages 266–275, 2001.

[3] M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Proc. of the 15th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO'95*, pages 15–28, 1995.

[4] L. Buttyan, P. Schaffer, and I. Vajda. RANBAR:RANSAC-based resilient aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2006.

[5] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2006.

[6] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proc. of IEEE Int'l Conf. on Data Engineering (ICDE)*, 2004.

[7] Funda Ergun, Sampath Kannan, S. Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *JCSS*, 60:717–751, 2000.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[9] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[10] Grubbs Frank. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):121, 1969.

[11] A. Gabrieli, L. Mancini, S. Setia, and S. Jajodia. Securing topology maintenance protocols for sensor networks: Attacks & countermeasures. In *SecureComm 2005. First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005.* IEEE, 2005.

[12] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for highly integrity sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, Washington, DC, 2004.

[13] M. Garofalakis, J. M. Hellerstein, and P. Maniatis. Proof sketches: Verifiable in-network aggregation. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*.

[14] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315, 2003.

[15] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli. Fault tolerance techniques in wireless ad-hoc sensor networks. In *Sensors 2002, Proceedings of IEEE*, pages 1491 – 1496.

[16] S. Madden, M. J. Franklin, J.M. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad hoc sensor networks. In *Proc. of 5th USENIX Symposium on Operating Systems Design and Implementation*, 2002.

[17] Samuel Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of Intl. Conf. on Data Engineering*, pages 555–566. IEEE Computer Society, 2002.

[18] A. Manjhi, S. Nath, and P. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *Proc. of ACM International Conference on Management of Data (SIGMOD)*, 2005.

[19] James Reserve Microclimate and Video Remote Sensing. http://www.cens.ucla.edu.

[20] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004.

[21] Habitat Monitoring on Great Duck Island. http://www.greatduckisland.net/.

[22] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2005.

[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Seventh Annual International Conference on Mobile Computing and Networks (MobiCOM)*, 2001.

[24] Adrian Perrig, John A. Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[25] The Firebug Project. http://firebug.sourceforge.net.

[26] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003.

[27] Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2006.

[28] A. Silberstein, R. Braynard, C. Ellis, K. Munagala, and J. Yang. A sampling-based approach to optimizing top-k queries in sensor networks. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*.

[29] D. Wagner. Resilient aggregation in sensor networks. In *Proc. of ACM Workshop on Security of Sensor and Adhoc Networks (SASN)*, 2004.

[30] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proc. of ACM MOBIHOC*, 2006.

[31] Y. Yao and J. E. Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(2):9–18, September 2002.

[32] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Proc. of IEEE Infocom*, 2004.

[33] W. Zhang and G. Cao. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration-based approach. In *Proc. of IEEE Infocom*, 2005.

[34] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003.

[35] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring sensor networks. In *Proc. of the 2nd IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[36] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering injected false data in sensor networks. In *Proc. of IEEE Symposium on Security and Privacy*, 2004.

[37] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks*, 2(4):500–528, 2006.