SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks[*]

Yi Yang, Xinran Wang, Sencun Zhu[†] and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {yy5,xinrwang,szhu,gcao}@cse.psu.edu

ABSTRACT

Hop-by-hop data aggregation is a very important technique for reducing the communication overhead and energy expenditure of sensor nodes during the process of data collection in a sensor network. However, because individual sensor readings are lost in the per-hop aggregation process, compromised nodes in the network may forge false values as the aggregation results of other nodes, tricking the base station into accepting spurious aggregation results. Here a fundamental challenge is: how can the base station obtain a good approximation of the fusion result when a fraction of sensor nodes are compromised?

To answer this challenge, we propose SDAP, a Secure Hop-by-hop Data Aggregation Protocol for sensor networks. The design of SDAP is based on the principles of *divide-and-conquer* and *commit-and-attest*. First, SDAP uses a novel probabilistic grouping technique to dynamically partition the nodes in a tree topology into multiple logical groups (subtrees) of similar sizes. A commitment-based hop-by-hop aggregation is performed in each group to generate a group aggregate. The base station then identifies the suspicious groups based on the set of group aggregates. Finally, each group under suspect participates in an attestation process to prove the correctness of its group aggregate. Our analysis and simulations show that SDAP can achieve the level of efficiency close to an ordinary hop-by-hop aggregation protocol while providing certain assurance on the trustworthiness of the aggregation result. Moreover, SDAP is a general-purpose secure aggregation protocol applicable to multiple aggregation functions.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; D.4.6 [**Operating Systems**]: Security and Protection—*Cryptographic controls*; K.6.5 [**Management of Computing and Information Systems**]: Communication Networks—*Security and Protection*

General Terms

Security, Algorithm, Design

Keywords

Data Aggregation, Probabilistic Grouping, Commit-and-Attest, Hop-by-hop, Sensor Network Security

[*]This work was supported in part by Army Research Office (W911NF-05-1-0270) and the National Science Foundation (CNS-0524156).

[†]Also with College of Information Sciences and Technology, PSU

1. INTRODUCTION

Wireless sensor networks are envisioned to be economic solutions to many important applications, such as real-time traffic monitoring, military surveillance, and homeland security [1]. A sensor network may consist of hundreds or thousands of low-cost sensors, each of which acts as an information source, sensing and collecting data from the environment for a given task. In addition, there may also exist one or more base stations (or data sinks) which subscribe to specific data streams by distributing interests or queries. The sensors in the network then push relevant data to a querying base station (BS). However, it is very inefficient for every sensor node to report their raw data back because every data packet need traverse many hops to reach the BS and especially sensor nodes are often constrained by scarce resources in memory, computation, communication, and battery. On the other hand, as in many cases sensor nodes in an area detect the common phenomena, there is high redundancy in their raw data. Thus, reporting raw data is often unnecessary.

Recently many data aggregation protocols [2, 3, 4, 5, 6, 7, 8] have been proposed to eliminate the data redundancy in sensor data of the network, hence reducing the communication cost and energy expenditure in data collection. During a typical data aggregation process, sensor nodes are organized into a tree hierarchy rooted at a BS. The non-leaf nodes act as aggregators, fusing the data collected from their child nodes before forwarding the results towards the BS. In this way, data are processed and fused at each hop on the way to the BS, and communication overhead can be largely reduced.

Hop-by-hop aggregation, however, opens a new door to false data injection attacks. Sensor nodes are often deployed in open and unattended environments, so they are vulnerable to physical tampering due to the low manufacturing cost. An adversary can obtain the confidential information (e.g., cryptographic keys) from a compromised sensor and reprogram it with malicious code. The compromised node may then report an arbitrary false fusion result to its parent node in the tree topology, causing the final aggregation result to far deviate from the true measurement. This attack can become more damaging when multiple compromised nodes collude in injecting false data.

The above attack is extremely difficult, if not impossible, to prevent or detect. From the viewpoint of information theory, data aggregation is a lossy data compression process because all the individual sensor readings are lost in the per-hop aggregation process. Hence, it is impossible for the BS to verify the correctness of an aggregated result without knowing the original readings. Unfortunately, the requirement of knowing the original readings effectively precludes any data aggregation techniques. As such, in practice a tradeoff between efficiency and accuracy must be made. The challenge now becomes: how can the BS obtain a good approximation of the aggregation result without losing the efficiency of per-hop data aggregation when a fraction of sensor nodes are compromised?

To answer this challenge, we propose SDAP, a Secure Hop-by-hop Data Aggregation Protocol for sensor networks. The design of SDAP is motivated by the following observation. During a normal hop-by-hop aggregation process in a tree topology, (implicitly) we need to place more trust on the high-level nodes (i.e., nodes closer to the root) than the low-level nodes, because the aggregated result calculated by a high-level node is due to a larger number of sensor nodes. In other words, if a compromised node is closer to the root, the bogus aggregated data from it will have a larger impact on the final result computed by the root. However, in reality none of these low-cost sensors should be more trustable than others. As such, SDAP takes the approach of reducing the trust on high-level nodes, which is realized by the principle of *divide-and-conquer*. More specifically, by using a probabilistic grouping method, SDAP *dynamically* partitions the topology tree into multiple *logical* groups (subtrees) of similar sizes. Since fewer nodes will be under a high-level node in a logical subtree, the potential security threat by a compromised high-level node is reduced.

To preserve the efficiency of per-hop aggregation, SDAP performs hop-by-hop aggregation in each logical group and generates one aggregate from each group. In addition, based on the principle of *commit-and-attest*, SDAP enhances an ordinary hop-by-hop aggregation protocol with commitment capability, which ensures that once a group commits its aggregate this group cannot deny it later. After the BS has collected all the group aggregates, it then identifies the suspicious groups based on a bivariate multiple-outlier detection algorithm. Finally, each group under suspect participates in an attestation process to prove the correctness of its group aggregate. The BS will discard the individual group aggregate if a group under attestation fails to support its earlier commitment made in the collection phase; the final aggregate is calculated over all the group aggregates that are either normal or have passed the attestation procedure.

Our analysis and simulations show that SDAP can achieve the level of efficiency close to an ordinary hop-by-hop aggregation protocol while providing certain assurance on the trustworthiness of the aggregation result. Unlike the trimming-based resilient aggregation [9] that simply ignore some fraction of highest and lowest values without any reasoning, our attestation scheme provides effective means to validate and then probably accept the abnormal values, as oftentimes we are more interested in those abnormal values than normal ones. As such, there is zero false positive in SDAP. Moreover, SDAP is a general-purpose secure aggregation protocol applicable to multiple aggregation functions.

The remainder of this paper is organized as follows. Section 2 describes our system model and design goals. In section 3, we propose our secure data aggregation protocol composed of grouping, aggregation and attestation. Security analysis and performance evaluation of our scheme are presented in section 4 and section 5, respectively. After that, section 6 describes related work in literature. Finally, we summarize our work and discuss the future work in section 7.

2. SYSTEM MODEL AND DESIGN GOALS

This section describes our system model and design goals, followed by the notations used in the description of the protocol.

2.1 Network Model and Key Setup

Network Model We assume a sensor network consisting of a large number of resource-limited sensor nodes (e.g., MICA motes [10]). In addition, there exists a powerful BS that connects the sensor network to the outside infrastructure such as the Internet. As in other data aggregation protocols [6, 11], we assume a topological tree rooted at the BS. There are various methods for constructing the aggregation tree according to different application requirements. However, SDAP does not rely on a specific tree construction algorithm as long as there is one. To concentrate on the security aspects of data aggregation, we will not address the general issues regarding data aggregation, e.g., what sensor applications might benefit from the technique of data aggregation or how to ensure time synchronization among nodes.

In a real application, a topology tree may be dynamic due to node or link failures. In TinyOS [12], a beaconing message is flooded every 30 seconds to reconstruct the broadcast tree. Clearly, it will be too costly for the BS to keep track of the network topology for every topology change, because every topology discovery may require every node to report its parent/child information to the BS. As such, in our scheme, we assume that the BS does not know the shape of the tree and its distance (in number of hops) from every node although it may want to discover the tree topology occasionally for other purposes.

We also assume there is a reliable transmission mechanism, for example, by using a link-layer hop-by-hop acknowledgment protocol. Thus, the various types of packets in our scheme will not be lost.

Key Setup We assume the BS cannot be compromised and it has a secure mechanism (e.g., μTESLA [13]) to authenticate its broadcast messages to all the nodes in the tree and every node can verify the received broadcast messages. We also assume every sensor node has an individual secret key shared with the BS. Further, there is a unique pairwise key shared between each pair of neighboring nodes [14, 15, 16].

2.2 Attack Model

Since a standard authentication primitive , e.g., message authentication code (MAC)s, can be employed to easily defeat an outsider adversary (who do not have any authentication keys) from launching many attacks, we assume an adversary can compromise a (small) fraction of sensor nodes to obtain the keys as well as reprogram these sensor with attacking code. There may be multiple potential attacks against a tree-based aggregation protocol. One type of attacks is behavior-based, in which the goal of an attacker is to disrupt the normal operation of the sensor network. For example, once a sensor node in the tree is compromised, it can attack the underlying routing protocol, drop other nodes' readings on purpose, or cause denial of message attacks [17] to deprive other nodes from receiving broadcast messages of the BS.

In this paper, however, we are not addressing any of these behavior-based attacks; instead, we focus on defending against *false data injection attacks* where the goal of an attacker is to make the BS to accept false sensor reports. In many situations, values received by the BS provide a basis for critical decisions; hence, false or biased values may cause catastrophic consequences. For example, when forwarding other sensor nodes' reported values, a compromised node may modify their values; it may also forge some false sensor readings on its own behalf. Because the measurements of the physical world are inherently noisy, if an attacker forges sensor readings that have negligible influence on the final aggregation result, he gains little. Therefore, we assume that an attacker aims to inject false values that deviate from the true measures in a noticeable scale. Apparently, the attacker does not want to be detected when launching this attack.

In particular, in the context of data aggregation, an aggregate usually contains not only a data value computed for the required aggregation function but also a count value indicating the number of sensor nodes involved in the aggregation operation. Clearly, an attacker can forge an unusual false data value as well as a large count value to make its false data account for a large portion of the final aggregated result. We refer to these two types of attacks *value changing attack* and *count changing attack*, respectively.

Next we show through an example why value changing attack and count changing attack are severe attacks. Suppose the BS queries the network for the average temperature and any sensed value must be between 32F and 150F. Let us assume a compromised node receives from its child nodes the aggregated data 100F and the count value 50. If the compromised node cannot modify the received aggregate, i.e., it can only forge a false reading of its own, then the aggregation data may range from 98.7F($\frac{100*50+32}{51}$) $\sim$ 101F($\frac{100*50+150}{51}$), which does not deviate far away from the true average value. However, if it can launch a count changing attack by reporting a bogus large count value, then it can make the average result be any value in the range from 100F to 150F (assuming its own reported temperature is 150F). Similarly, if the compromised node can launch a value changing attack by modifying the data value in its child nodes' aggregate, it can easily make the average result be either 150F or 32F as desired. Obviously, if possible, an attacker can combine and launch these two attacks simultaneously to affect the final aggregate without being detected.

Note that we do not consider the attack where a compromised node forges a false reading of its own as a value changing attack. First, as we shown in the above example, the impact of such an attack is usually limited. Second, such a compromised node is very much like a faulty sensor node. In this case, we have to rely on an outlier detection algorithm or the content-based attestation scheme proposed in Section 3.4.4.

2.3 Design Goal

Our design goal is to defend against the false data injection attacks making the BS accept false aggregation results, and we will focus on two kinds of false data injection attacks, value changing attacks and count changing attacks. Specifically, our design goal includes:

- *Low communication overhead*: The purpose of conducting aggregation is to reduce communication overhead. Clearly if the overhead of our scheme is equivalent to that of a raw data based scheme, there is no need to employ our scheme.
- *Effectiveness*: The BS should have a high probability to detect the injected false values. Once false values are detected, they will be discarded. This is important to ensure the accuracy of the final aggregation result.
- *Generality*: Since it is undesirable to design one scheme for one aggregation function, our scheme should apply to various aggregation functions, such as MAX/MIN, MEAN, SUM, COUNT, and so forth.

Notations The following notations are used in the description of the protocol:

- BS refers to the BS. u, v, w, x, y are principals, i.e., the identifiers of sensor nodes.
- $K_{u,v}$ is the pairwise key shared between node u and node v, and K_u is the individual key shared between node u and the BS.
- $m1|m2$ denotes the concatenation of two messages $m1$ and $m2$.
- $E(K, m)$ refers to the encryption of message m using key K.
- $MAC(K, m)$ is the message authentication code (MAC) of message m with key K.

In addition, we will use $u \rightarrow v : M$ to denote a one-hop delivery of message M from u to a neighbor v and $u \rightarrow\rightarrow v : M$ to denote a delivery that may involve multiple hops.

3. THE SECURE DATA AGGREGATION PROTOCOL

In this section, we present our Secure Data Aggregation Protocol (SDAP). We first give an overview of the protocol and then present the details of the protocol.

3.1 Protocol Overview

The design of SDAP is based on the principles of *divide-and-conquer* and *commit-and-attest*. First, SDAP uses a novel probabilistic grouping technique to partition the nodes in a tree topology into multiple logical groups (subtrees) of similar sizes. A commitment-based hop-by-hop aggregation is performed in each group to generate a group aggregated result. The BS then identifies the suspicious groups based on the set of group aggregated results. Finally, each group under suspect participates in an attestation process to prove the correctness of its group aggregate. Next, we present the details of the protocol, which includes three phases: *query dissemination*, *data aggregation*, and *attestation*.

3.2 Tree Construction and Query Dissemination

For concreteness, we first describe a simple aggregation tree construction algorithm, which is similar to that in [6]. Initially, the root broadcasts a tree construction beaconing message which includes its own id and its depth to be 0. When a node, say x, receives a broadcast message at its first time from a node y, x assigns its depth to be the depth of y plus one, and its parent to be y. After this, it rebroadcasts the message. This process continues until all nodes have received this message.

After constructing the aggregation tree, the BS can disseminate the aggregation query message through this tree. Besides the aggregation function that represents the BS's request, a random number is added to the query. This random number is generated by the BS as a grouping seed, which is used for the probabilistic grouping in the next phase. Specifically, a query packet that the BS broadcasts is as follows:

$$BS \rightarrow\rightarrow * : F_{agg}, S_g$$

where F_{agg} refers to a specific aggregation function, such as MIN /MAX, MEAN, SUM, and S_g is the random number generated for each query. We may employ μTESLA [13] to provide global broadcast authentication of the query dissemination.

Above we discussed query dissemination after tree construction to make it independent of the tree construction protocol. In practice, we may combine these two steps into one. The query information can be piggybacked in a beaconing message. On the other hand, the dissemination of a query can help reconstruct the tree topology, thus mitigating the tree partition problem due to node or link failures.

3.3 Probabilistic Grouping and Data Aggregation

Through the previous phase, all nodes have identified their parents. In this phase, SDAP randomly groups all the nodes into multiple logical groups and performs aggregation in each group. Probabilistic grouping is conducted through the selection of leader node for each group. During the aggregation, every node makes its commitment by embedding some security information to its aggregate. Next, we first describe how group leaders are selected, and then discuss techniques to add security information into the aggregated data.

3.3.1 Group Leader Selection

Group leaders are selected on-the-fly based on the count values and the grouping seed S_g received in the query dissemination phase. Two functions are used in group leader selection. One is a cryptographically secure pseudo-random function H that uniformly maps the input values (node id and S_g) into the range of $[0, 1)$; the other is a grouping function F_g that takes a positive integer (count) 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

$$H(S_g|x) < F_g(c) \tag{1}$$

where c is the count value of node x (we will see how c is calculated in the next subsection). If this inequation 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. An example of a grouped tree is shown in Figure 1. Here x is a group leader and the nodes included in the dashed line are its group members. Similarly, w'' is another group leader.

The grouping function F_g is used to control the probability for a node to be chosen as a group leader and it is preloaded in each sensor. Because the output of H is uniformly distributed between 0 and 1, the probability that it is smaller than $F_g(c)$ actually equals to the value of $F_g(c)$. In our construction, $F_g(c)$ increases with the count value c. Thus, if a node has a larger count value, the probability for it to become a leader is higher. By adjusting the grouping function, ideally, the resulted group sizes are roughly even with a small deviation, which provides the basis for our attestation. A specific grouping function is selected and the grouping result is shown in Section 5.1.

The use of the random number S_g as the grouping seed is mainly for security reasons. With the random number, the BS can rotate the leaders among nodes instead of fixing their roles, so that the attackers cannot determine in advance which nodes will be the group leaders for each query. Otherwise, the attacker may target at the group leaders and compromise them. Also, because a different S_g is used each time, every node is assigned into a different group that is formed on the fly. This helps thwart some prearranged colluding attacks by multiple compromised nodes. Another advantage is to balance the resource usage of nodes (e.g., storage, computation, and communication) so as to prolong the overall lifetime of the network.

3.3.2 Aggregation Commitment

Before describing the data aggregation process, we first introduce the packet format used in the commitment. Each aggregation packet contains the sender's id, an aggregated data value, and a count value to indicate how many nodes contributing to the aggregated data. In addition, a flag field (one bit) is contained in each packet to show whether the aggregate needs to be aggregated further by the nodes enroute to the root. Flag value '1' means that no further aggregation is needed, whereas '0' means to be aggregated. This flag field is initialized to '0'. After a group leader finishes the aggregation for the group, this flag field is set to '1', and other nodes on the path to the root just forward those packets with flag '1'.

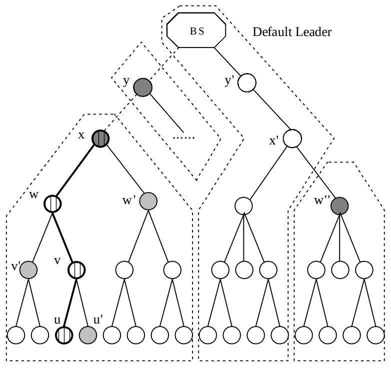

Figure 1: An example of the aggregation tree. The nodes x, y and w" with the color dark gray are leader nodes, and the BS as the root is a default leader

The pairwise key shared between each pair of parent and child is used to encrypt the aggregate. This encryption in practice provides not only confidentiality but also authentication. This is because the content format is known to the BS and the value of each item should fall in a certain range. Thus, using encryption saves the bandwidth that will otherwise be used for an additional message authentication code (MAC). In addition, a MAC computed using the key shared with the BS is also attached at the end of each packet, which provides authentication to the BS. Next, we present details of the aggregation process.

Leaf node aggregation: Different from query dissemination, data aggregation starts from the leaf nodes towards the BS. Since a leaf node does not need to do aggregation, it just sends its id, data and count value to its parent (it also keeps a local copy until the attestation phase is completed). The following shows the packet that a leaf node u sends to its parent v:

$$\begin{aligned} u \rightarrow v \quad : \quad & u, 0, E(K_{u,v}, 1|R_u|S_g)|MAC_u \\ & MAC_u = MAC(K_u, 0|1|u|R_u|S_g) \end{aligned}$$

where '0' is the aggregation flag, '1' is the count value, R_u the reading of node u, and MAC_u the MAC value computed by node u with its individual key. Here S_g is included to identify the query as well as to prevent replay attacks.

Intermediate node aggregation: When an intermediate node receives an aggregate from its child node, it first checks the flag. If the flag is '0', it keeps a local copy of the aggregates (until the attestation phase is done) and performs further aggregation; otherwise, the node directly forwards the packet to its parent node.

More specifically, for a report with flag '0' received from a child node, a node first decrypts the data using its pairwise key shared with this child node. It also performs some simple checking on the validity of the count, R_u (if within a certain range), and S_g (if the same as the one received in the query dissemination phase). If the aggregate packet does not pass this checking, it will discard the packet. Otherwise, it will further aggregate its own reading with all the aggregates received from its child nodes with flag '0'. A new count is also calculated as the sum of the count values in the received aggregates with flag '0' plus one (considering its own reading). The node checks if it is a group leader based on the same inequation (1) using its own id and the new count as the inputs. The node then encrypts the new count value and aggregation data using the pairwise key shared with its own parent.

As shown in Figure1, w is the parent of v. Since here node v is not a leader. The packet that v sends to w is as follows:

$$v \rightarrow w : v, 0, E(K_{v,w}, 3|Agg_v|S_g)|MAC_v$$
$$Agg_v = F_{agg}(R_v, R_u, R_{u'})$$
$$MAC_v = MAC(K_v, 0|3|v|Agg_v|MAC_u \oplus MAC_{u'}|S_g)$$

where '3' is the count value summed over the count value of u, u' and its own contribution, Agg_v is the aggregation value of node v and MAC_v is the MAC value computed by node v. Note that the MAC of an intermediate node is calculated over not only the previous fields but also the XOR of the MACs from its children. In this way, a MAC value is also computed in a hop-by-hop fashion, thus it can represent the authentication information of all the nodes contributing to the data.

Leader node aggregation: Now suppose that an intermediate node has processed the aggregates from its child nodes and it finds out that it is a group leader based on (1). Like a regular intermediate node, it also computes a new aggregate, keeps local copies of those packets with flag '0', and appends a corresponding MAC using its individual key shared with the BS. Unlike a regular intermediate node, it encrypts the new aggregate with its individual key and sets the flag to '1' in its aggregation packet. Since in Figure 1 node x is a group leader, the packet it sends upward is as follows:

$$x \rightarrow\rightarrow BS : x, 1, E(K_x, 15|Agg_x|S_g)|MAC_x$$
$$Agg_x = F_{agg}(R_x, Agg_w, Agg_{w'})$$
$$MAC_x = MAC(K_x, 1|15|x|Agg_x|MAC_w \oplus MAC_{w'}|S_g)$$

where Agg_x is the aggregation result of the group and MAC_x is the MAC value computed by the leader node x. Note that the leader node needs to set the flag field to '1', so that data from this group will not be aggregated any more. That is, in Figure 1, when node y receives a packet from x, it forwards the packet towards the BS without any further aggregation and it does not add the count value of x to its own. In an extreme case when all the children of a node are group leaders, this node will only contribute count value 1 to its parent node, similar to a leaf node. As such, we can see that the importance of a higher level node is reduced as we have desired.

Based on the above aggregation rule, the aggregated data and the corresponding MACs are transmitted to the BS. There may be some nodes left without group membership. In this case, the BS is the default group leader for them.

After the BS receives the aggregates from all groups, it decrypts and saves them in the following format: (x, c_x, Agg_x, MAC_x), where x is the leader node's id, c_x is the group size, Agg_x is the group's aggregation result and MAC_x is the authentication tag computed by the group leader. Note that all the groups are logical groups; no physical partition of the topology tree is involved.

We notice that although the spirit of this technique is similar to Merkle hash tree [18], there are several noticeable differences. First, Merkle hash tree is a data structure, not a real topology tree; second, Merkle hash tree is a binary tree whereas in our case the topology tree is arbitrary. Third, in Merkle hash tree only leaves are measurements, all others are hash values. Fourth, the value of a MAC in our scheme is computed over more information.

3.3.3 *Tracking the Forwarding Path*

When a sensor node receives an aggregation packet with flag '1', it records into its forwarding table the following information: S_g, the id of the group leader, the incoming link (i.e., from which node it receives the packet). In this way, when the BS sends out an attestation request later regarding this group, the node knows where to forward this request. This can save some message overhead because otherwise the BS has to flood the request. For example, as shown in Figure1, when node y receives the packet from x, it forwards the packet to the BS and adds x to its forwarding table. In the future, if the BS wants to attest the group of x, it sends the attestation message directly to its child y. Since x is in y's forwarding table, y also forwards this attestation message directly to x.

The above solution works fine in most cases. If the aggregation tree is very large and there are many groups, techniques such as Bloom filters [19] may be used to reduce the storage overhead. We note that the size of a forwarding table does not necessarily keep increasing because it is updated in each query.

3.4 Verification and Attestation

3.4.1 *Verifying the Aggregation Messages*

After the BS has received the aggregation messages from the group leaders, it needs to verify the authenticity of the aggregated value in each aggregation message. This includes verifying the content of the packet and the authenticity of the leader. First, based on the group leader id, say x, in the message, the BS can find out the individual key of the node (K_x) by which it decrypts the data and gets the following information: the count value c_x, the aggregated value Agg_x, and S_g. The authenticity of the message is provided because the content format is known to the BS and the value of each item should fall in certain range. Second, the BS verifies the legitimacy of the claimed group leader x by checking whether $H(S_g|x) < F_g(c_x)$ because the BS knows H, F_g and the grouping seed S_g. If this does not hold or any item in the packet is invalid, the BS simply drops the packet.

3.4.2 *Determining Suspicious Groups for Attestation*

After the above verification, the BS believes about the aggregate, say (c_x, Agg_x), is truly from a legitimate leader x. However, the BS cannot tell whether c_x or Agg_x has been modified because a compromised group node or the leader x may have modified the data, which can influence the final aggregation result at the BS. Note that authentication cannot solve this insider attack because a compromised node has the valid keys.

We expect the attacker to forge an aggregated data that have a non-trivial influence on the final result; otherwise the attacker could not gain much. As a result, a false aggregate should exhibit certain abnormality. On the other hand, we cannot simply treat all abnormal sensing data as outliers and discard them, since they may indeed reflect the real environment. In many cases we are more interested in abnormal data than in normal ones. For example, for sensors deployed to detect fire events, abnormally high temperature is our special interest. With these in mind, we have to verify the abnormal aggregates before accepting or rejecting them. In other words, the BS should attest the groups with suspicious large count values or doubtful aggregation data.

We mainly use Grubbs' test [20], also known as the maximum normalized residual test, to detect the outliers. However, we also make several modifications so that it can detect *multiple* outliers from *bivariate* data. In Grubbs' test, the Null hypothesis H_0 means that there are no outliers in the data set, whereas the hypothesis H_1 means that there are at least one outliers in the data set. More specifically, it first computes the sample statistic for each datum χ

Algorithm 1 Outlier Detection Algorithm

```
Input: a set $T$ of $n$ tuples $(x, c_x, Agg_x)$, where $x$ is group leader
id, $c_x$ is group count value, $Agg_x$ is group aggregation result, and
$n$ is total number of groups;
Output: a set $L$ of leader ids of groups with outliers;
Procedure:
 1: loop
 2:     compute mean $\mu_c$ and standard deviation $s_c$ for all the
        counts in set $T$;
 3:     compute mean $\mu_v$ and standard deviation $s_v$ for all the val-
        ues in set $T$;
 4:     find the maximum count value $c_x$ in set $T$;
 5:     compute statistic $Z_c$ for count $c_x$: $\frac{c_x - \mu_c}{s_c}$;
 6:     compute $p$-value $P_c$ based on the statistic $Z_c$;
 7:     compute statistic $Z_v$ for the corresponding value $Agg_x$:
        $\frac{|Agg_x - \mu_v|}{s_v}$;
 8:     compute $p$-value $P_v$ based on the statistic $Z_v$;
 9:     if $(P_c * P_v) < \alpha$ then
10:         $T = T - \{(x, c_x, Agg_x)\}$;
11:         $L = L \bigcup \{x\}$;
12:     else
13:         break;
14:     end if
15: end loop
16: return $L$;
```

in the set by $\frac{|\chi-\mu|}{s}$, where μ and s are the sample mean and standard deviation of all the data, respectively. The result represents the datum's absolute deviation from the sample mean in units of the sample standard deviation. Based on this, there are two equivalent methods to decide whether H_0 should be accepted or not. One is to check whether the sample statistic falls in the non-rejection range defined by the critical values. The other one is to compare the p-value computed based on the sample statistic with the predefined significance level α (equals to 0.05 typically), where the p-value is the observed level of significance, defined as the probability that the sample statistic is equal to or more than the result obtained from the sample data given that H_0 is true. The smaller the p-value is, the farther the sample statistic deviates from the sample mean. When the p-value is smaller than α, H_0 is rejected and the datum under consideration is an outlier.

When we apply Grubbs' test in our setting, we need to make several extensions. First, since Grubbs' test detects one outlier at a time, we can expunge the detected outlier from the dataset and iterate the test over the remaining data until no outliers can be found. In this way we can detect multiple outliers. Second, Grubbs' test is normally used for univariate data set, but we will need to detect outliers from bivariate data (i.e., counts and aggregation data). Because counts and data are independent variables, we set the p-value as the product of these two p-values to prevent an attacker from either forging a large count or an extreme value. Normally, a datum of one variable is considered to be an outlier when its p-value is smaller than 0.05. In our bivariate case, even when each separate one is less like an outlier, we may still consider it as an outlier. For instance, for a count and data value pair reported by a group, we may get the p-value 0.2 for the count and 0.24 for the data. None of them is smaller than 0.05; however, their product is $0.048 < 0.05$. Thus, we identify this group as a suspicious group. In another example, to avoid detection an attacker may report a very small count value but extreme data. In this case, we may get the p-value of 1.0 for the count, but as long as p-value for the data is less than 0.05, this group will still be selected. A formal description of the outlier detection algorithm is shown in Algorithm1.

We have seen that an attacker does not have much motivation for forging a small count. As such, we are only interested in large count values. That is, for count values, the BS will run the one-sided Grubbs' test for computing the p-values. For data values, we may consider a two-sided test for some aggregation applications, such as MEAN. For other operations such as MIN/MAX, it depends on the specific aggregation functions. We may also resort to the content-based attestation introduced in Section 3.4.4 to deal with these data outliers.

3.4.3 Generating and Forwarding Attestation Requests

After the BS has decided which group(s) to attest, it will need to decide how to attest the group. The challenge is due to the fact that the BS only knows the group leader id — it does not know what the other nodes are and how they form the group topology. In this case, how can it prevent the group leader from making up the group topology and attested results? Next we show a simple while effective way to address this challenge.

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 as shown shortly. S_g is included for identifying the query. Let x be the id of the leader node for the group to be attested and y the id of the node from which the BS received the group aggregate (BS also maintains a forwarding table).

$$BS \rightarrow y : x, S_a, S_g \qquad (2)$$

Again, we can use μTESLA to provide broadcast authentication.

The attestation request from the BS will be disseminated down the tree. Every node receiving this request searches its forwarding table using the leader id as the index to get the next-hop node id. It then forwards the request to that next-hop node.

3.4.4 Group Attestation

During a group attestation process, a physical attestation path between the group leader and a leaf node (in the group subtree) is dynamically formed. More specifically, after the leader node receives the attestation request from the BS, it decides the next hop on the attestation path as follows. It first adds up all the count values of its child nodes in the logical group (not all the child nodes in the physical tree because some child nodes may become group leaders themselves) and calculates $\sum_{k=1}^{d} c_k$, where c_k is the count value of its k^{th} child and d is the number of its children in the group. This can be done since the parent node stored all the count values from the children nodes in the aggregation phase. Then, it calculates $\sum_{k=1}^{d} c_k \cdot H(S_a|id)$ for each of its children id based on the pseudo-random function H. The parent picks up the i^{th} child for attestation if the calculated value falls in the interval $[\sum_{k=1}^{i-1} c_k, \sum_{k=1}^{i} c_k)$. We will prove in the next subsection that this construction ensures that the probability for a child node to be selected on the path is proportional to its count value reported in the aggregation phase. Thus, a child with a larger count will be attested with a higher probability[1].

A selected child runs the same process to select one of its own children to form the path. Recursively, an attestation path between the leader and a leaf node (in the logical group subtree) is formed. Each node on the path sends back 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, aggregation data, and their MACs.

[1]Instead of using count values only, a variant of this idea is to use some function (e.g., multiplication) of count and data as the criteria to detect a compromised node that forges small count but extreme data.

Figure 1 shows one example. Assume that the BS wants to attest the group with leader node x and the attestation path in this group is x−w−v−u. Then, the messages sent back to the BS from this group are:

$$\begin{aligned}
x \rightarrow\rightarrow BS &: x, E(K_x, x|15|R_x) \\
w \rightarrow\rightarrow BS &: w, E(K_w, w|7|R_w) \\
w' \rightarrow\rightarrow BS &: w', E(K_{w'}, w'|7|Agg_{w'}|MAC_{w'}) \\
v \rightarrow\rightarrow BS &: v, E(K_v, v|3|R_v) \\
v' \rightarrow\rightarrow BS &: v', E(K_{v'}, v'|3|Agg_{v'}|MAC_{v'}) \\
u \rightarrow\rightarrow BS &: u, E(K_u, u|1|R_u) \\
u' \rightarrow\rightarrow BS &: u', E(K_{u'}, u'|1|R_{u'})
\end{aligned}$$

These messages are encrypted by the individual keys of the sensor nodes.

After the BS decrypts the received data, it first verifies whether w, v and u are really the nodes on the attestation path based on S_a, these nodes' ids, and the counts. Then, 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 by itself and reconstructs the aggregation result of this group, Agg_x, to examine whether nodes on the path have forged the aggregation results in the aggregation phase:

$$\begin{aligned}
Agg_v &= F_{agg}(R_v, R_u, R_{u'}) \\
Agg_w &= F_{agg}(R_w, Agg_v, Agg_{v'}) \\
Agg_x &= F_{agg}(R_x, Agg_w, Agg_{w'})
\end{aligned}$$

It can also reconstruct MAC_x using these data:

$$\begin{aligned}
MAC_u &= MAC(K_u, 0|1|u|R_u|S_g) \\
MAC_{u'} &= MAC(K_{u'}, 0|1|u'|R_{u'}|S_g) \\
MAC_v &= MAC(K_v, 0|3|v|Agg_v|MAC_u \oplus MAC_{u'}|S_g) \\
MAC_w &= MAC(K_w, 0|7|w|Agg_w|MAC_v \oplus MAC_{v'}|S_g) \\
MAC_x &= MAC(K_x, 1|15|x|Agg_x|MAC_w \oplus MAC_{w'}|S_g)
\end{aligned}$$

Note that here some of the reconstructions may not be necessary. For example, if the BS compares the reconstructed aggregation result with the previously received one and finds that they are not consistent, there is no need for the BS to recompute the MAC value. Only when both the aggregation result and the MAC value match the previously received commitment, the BS accepts the data and use them to compute the final aggregation result. Otherwise, the BS knows that some node in this group has been compromised and it discards this group aggregate.

Attesting Multiple Paths The above technique is for one path attestation. To improve the detection capability, we may select multiple paths for attestation. One straightforward solution is to send multiple attestation seeds, each of which is used to determine one path. A more efficient way is as follows. In its attestation request the BS adds n_g, the number of paths to be attested. When a group node selects its child nodes, it evaluates $H(S_a|id|k)$, where $k = 1, 2, ..., n_g$. Clearly, these multiple paths will overlap; if a node appears in multiple paths, it only needs to send back one report. Thus, the cost of attestation is sublinear with respect to the number of attested paths.

Other Attestation Techniques The above attestation is actually a depth-based one because it picks up attestation paths in a group. Alternatively, we may use a breadth-based scheme, in which the BS may ask, for example, all the nodes one or two levels below the group leader to supply their aggregates. This approach is good for attesting a more balanced tree because the higher level nodes usually have larger counts than the lower level nodes. However, it may not be effective for arbitrary tree topology. Also, it is more vulnerable to colluding attacks by several topologically consecutive compromised nodes.

Some aggregation functions are found inherently insecure, such as MIN/MAX, because the change to a single sensor reading can cause noticeable changes to the final result [9]. Therefore, we also consider a content-based attestation approach to validate an outlier. Specifically, once the BS notices such an outlier, it requests the sensor readings from the neighbors of this node and compares them. Since these nodes are close to each other, their readings should bear certain spatial or temporal correlation. Based on this knowledge, the BS can decide whether to accept the outlier or not. Since this technique is orthogonal to the presented techniques, we will study it in the future.

4. SECURITY ANALYSIS

This section first discusses how SDAP prevents several general attacks, then presents the qualitative results on its detection capability.

4.1 General Security Analysis

Our commit-and-attest technique aims to ensure that once a group has committed its aggregation result, if being attested later, every involved node in the group has to report its original aggregate. Otherwise, the group attestation process will detect the attack by finding the inconsistency between the committed aggregate and/or MAC and the reconstructed aggregate and/or MAC. This technique is secure as long as we use a cryptographically secure MAC function such as HMAC, although we will not give a rigorous proof here. Readers are referred to Merkle hash tree [18] on this.

Due to our probabilistic grouping scheme, an attacker cannot selectively compromise nodes to ensure his optimal attacking strategy: for example, making multiple of the compromised nodes or no more than one to appear in the same group. Because grouping is a dynamic process, a node cannot know in advance whether it will become a group leader or which group it will belong to. Also, because the aggregates from all the groups are encrypted, a compromised node cannot know if its own aggregate will become an outlier by Grubbs' test. 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.

A data aggregation protocol is generally vulnerable to a potential *event suppression attack,* where a compromised node changes its aggregated value corresponding to a real abnormal event to a normal value, thus the BS might not notice the real event. However, our probabilistic grouping technique can greatly mitigate this attack because (1) the role of an attack node in an aggregation subtree (group) is not fixed (group leader is randomly selected and every node may become a leaf node in an aggregation subtree), and (2) some other nodes belonging to other aggregation subtrees may detect the real event and report it to the BS.

4.2 Detection Rate Analysis

Next we discuss the effectiveness of SDAP in detecting the value changing attack and count changing attack. To detect either of these attacks or a combination of them, the first step is to identify the suspicious groups. In last section we proposed to use Grubbs' test for this purpose (certainly other appropriate outlier detection algorithms may also be applied), thus the probability of an attacked group being selected is determined by the power of Grubbs' test.

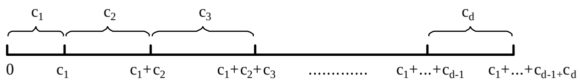

Figure 2: Count value intervals for a parent with d children

Note that Grubbs' test may identify an attack-free group as a suspicious one; however, from security viewpoint this is not an issue because this group will pass the group attestation anyway. In other words, our protocol has *zero false positive rate*.

Next we will focus on analyzing the probability of such an attack being detected by our group attestation scheme, given that the attacked group has been identified. For clarification, when we refer to a compromised node, we always assume that this node makes either count changing attack or value changing attack (a compromised node following our protocol is no different from a normal one).

4.3 Count Changing Attack Detection

For ease of presentation, let us first consider the case that there is only one compromised node in an attested group, although multiple compromised nodes, if they are in a logical group, may collude in launching attacks. A count changing attack will be detected when the compromised node is selected on an attestation path. In the following, we analyze the detection probability. We propose three lemmas that are related to this probability.

First, we derive the probability that a parent selects one child for attestation based on the selection rule introduced in Section 3.4.4.

LEMMA 1. *Suppose a parent has d children with counts c_1, c_2, ... , c_d respectively in a logical group. The probability that this parent selects the ith child with count c_i for attestation is $P(i,d) = \frac{c_i}{\sum_{k=1}^{d} c_k}$.*

PROOF. Because the value of $H(S_a|id)$ is uniformly distributed in the range $[0, 1)$, it can be treated as a random variable X that follows a uniform distribution with the pdf (probability density function)

$$f_X(x) = \begin{cases} 1, & \text{if } 0 \le x < 1 \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the value of $\sum_{k=1}^{d} c_k \cdot H(S_a|id)$ can be treated as another random variable Y uniformly distributed in $[0, \sum_{k=1}^{d} c_k)$, whose pdf is given by

$$f_Y(y) = \begin{cases} \frac{1}{\sum_{k=1}^{d} c_k}, & \text{if } 0 \le y < \sum_{k=1}^{d} c_k \\ 0, & \text{otherwise.} \end{cases}$$

From Figure2, we can see that the probability for a parent to select the ith child equals to

$$P(i,d) = \int_{\sum_{k=1}^{i-1} c_k}^{\sum_{k=1}^{i} c_k} \frac{1}{\sum_{k=1}^{d} c_k} dy = \frac{c_i}{\sum_{k=1}^{d} c_k},$$

which is proportional to the count value of this child. $\square$

Since an attestation path always starts from the leader node, if the leader node of a group made the count changing attack, the detection rate is 100%. Next we analyze the probability that a regular node is selected on the attestation path. We use the notation $c_{j,i}$ to denote the count value of a node in depth j (i.e., its distance from the leader node is j), which is also the ith child of its parent(Figure3). In this notation, $0 \le j \le h$ where h is the height of the group subtree and $1 \le i \le d_j$ where in depth j there are totally d_j children for selection. Therefore, the count value of the leader node is denoted by $c_{0,1}$. Count values of leader's children are denoted by $c_{1,1}$ to c_{1,d_1} from left to right. Suppose in depth j the attested node is denoted by u_j.

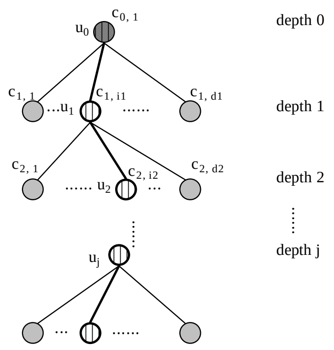

Figure 3: Choosing an attestation path in one group based on count values

LEMMA 2. *The detection rate of the attack by a compromised node u_j in depth j, i.e., the probability for node u_j to be selected on the attestation path, is $D_r(j) = \prod_{l=1}^{j} P(i_l, d_l)$.*

PROOF. When the sibling of node u_j's parent is selected, the probability to choose node u_j is 0; therefore, the probability of choosing node u_j with the parent u_{j-1} equals to the probability of choosing u_{j-1} multiplied by the probability of choosing u_j under the condition that we have selected the parent u_{j-1}.

According to Lemma 1, the probability for a child u_1 in depth 1 with count c_{1,i_1} to be selected on the path is $P(i_1, d_1)$, because the probability for us to choose the leader node is 100%. Hence, we have that the detection rate $D_r(j)$, i.e., the probability for node u_j to be selected on the attestation path, equals to

$$\begin{aligned} P(U_j) &= P(U_j|U_{j-1}) \cdot P(U_{j-1}) \\ &= P(U_j|U_{j-1}) \cdot P(U_{j-1}|U_{j-2}) \cdots P(U_1|U_0) P(U_0) \\ &= P(i_j, d_j) \cdots P(i_1, d_1) \cdot 1 \\ &= \prod_{l=1}^{j} P(i_l, d_l), \end{aligned}$$

where U_j refers to the event that u_j is selected on the attestation path. $\square$

Next we show the detection rate of the attack when we select multiple paths for attestation.

LEMMA 3. *Suppose we choose m independent attestation paths. The detection rate of the count changing attack by a compromised node u_j in depth j is $D_r(j,m) = 1 - (1 - D_r(j))^m$.*

PROOF. We can treat the selection of m attestation paths as m independent events. The probability of detecting a compromised node equals to the probability of selecting this node at least once in the m events. Suppose A refers to the event that node u_j is selected on the attestation path at least once in the m events. On the contrary, $\overline{A}$ means that node u_j is never selected on the attestation path in the m events. Based on Lemma 2, we have

$$P(\overline{A}) \quad = \quad (1 - \prod_{l=1}^{j} P(i_l, d_l))^m = (1 - D_r(j))^m.$$

Therefore, the detection rate $D_r(i, m)$ equals to

$$P(A) = 1 - P(\overline{A}) = 1 - (1 - D_r(j))^m.$$

$\square$

Since this is a function increasing with the value of m, we can see that if we perform the attestation for multiple times ($m > 1$), the detection rate will be higher, which means that we have more

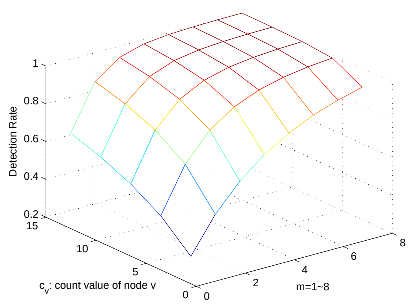

Figure 4: Detection rate of the count changing attack in the group with leader node x. For c_v=3,6,9,12,15, respectively. The number of attestation paths equals to $1 \sim 8$.

chances to detect the attack. Assume the node v in the group with leader x in Figure1 is an attacking node, our detection rate of the attack through multiple paths is shown in Figure4. For instance, if node v changes its count value from 3 to 6. Accordingly, the count values of node w and x becomes 10 and 18, respectively. If we choose only one attestation path, the detection rate of this attack is 39.22%, but if we choose four attestation paths, the detection rate is increased to 86.35%.

Finally, we consider the case when multiple compromised nodes are in the attested logical group. The detection rate is subject to the distribution of these compromised nodes, for example, whether more than one compromised nodes locate on a same path, no two nodes locate on a same path, or a hybrid of these two scenarios. If multiple compromised nodes are on the same attestation path, then the detection rate of the attack equals to the probability that the highest-level compromised node is selected; when all these compromised nodes are on different attestation paths, we can detect the attack as long as we can choose any of these paths, so the detection rate is the sum of the probability for choosing each one. For the hybrid case, the detection rate can be computed as the sum of the probability that we attest the highest-level compromised node in each of these paths. From the above analysis, we can see that if there are more than one compromised nodes in the attested group, the detection rate is higher unless these nodes are all on the same path.

4.4 Value Changing Attack Detection

Similar to a count changing attack, a value changing attack will be detected when the attacking node is selected on the attestation path. Because a compromised node may forge small count but extreme data to avoid the attestation, for detecting value changing attack, it is not effective to determine an attestation path only based on counts. Instead, we have to take into account the data value by using some function of count and data as the criteria to select a path. For example, we may simply replace the count values in the above path selection rule (Section 3.4.4) and lemmas with $c|R - R_{normal}|$, where c is a count, R is the corresponding data value, and R_{normal} is the normal data value (e.g., the normal indoor temperature). Of course, other appropriate functions can also be applied.

5. PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of SDAP. We first present a grouping function and show that it meets our requirements through simulated grouping results. Then, after we analyze the communication overhead of the protocol, we further use simulations to support our claim that SDAP only causes little extra overhead compared to hop-by-hop aggregation.

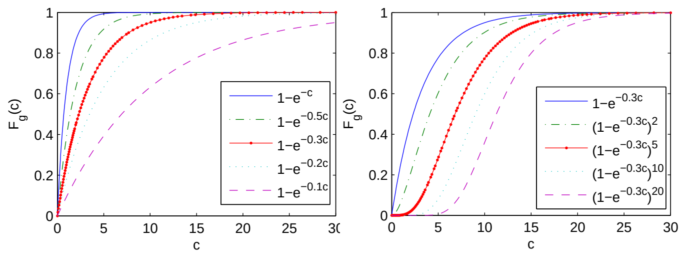

Figure 5: $F_g(c)$ as the function of count value c with parameters β, γ: $F_g(c) = (1 - e^{-\beta \cdot c})^\gamma$ $(0 < \beta \le 1, \gamma \ge 1)$

5.1 Grouping Function

In Section 3.3.1, a grouping function F_g was used to control the probability of a node being the group leader. This function generates an output value between 0 and 1, based on the count value. Our goal is to select a F_g which can ensure the group sizes are similar so as to reduce the variance. Hence, when the BS performs Grubbs' test, less likely a normal group will become an outlier for attestation, thus reducing the attestation overhead. Specifically, this grouping function should have the following requirements:

- if $c = 0$, $F_g(c) = 0$;
- if $c = 1$ (leaf node), $F_g(c) \approx 0$;
- if $c \to \infty$, $F_g(c) \to 1$, but $F_g(c) < 1$;
- the gradient of its curve increases slowly at first and decreases towards 0 after a peak value close to 1.

The first three requirements are apparent. Based on the fourth requirement, when the count value c is small, the probability of becoming a leader is low, whereas when the count value c is large enough, this probability is rapidly increased to a large value (e.g., larger than 50%). As a result, the group sizes becomes more similar.

To meet these requirements, we choose $F_g(c) = (1 - e^{-\beta \cdot c})^\gamma$ $(0 < \beta \le 1, \gamma \ge 1)$, where β is used to control the gradient of the curve and γ is used to control the shape of the curve (e.g., concave or convex). As shown in Figure5, as β increases, the curve becomes sharper. With a large γ, the function satisfies the fourth requirement of our grouping function.

5.2 Grouping Results

We verify that the grouping function satisfies our requirements through the simulated grouping results. In the simulation, 3000 nodes are randomly distributed in an area of $2000 * 2000 ft^2$. The transmission range is set to be $65ft$. Tree construction protocol introduced in 3.2 is used to build up the tree. For the grouping function, β and γ are set to 0.15 and 30, respectively. The simulation is run 5000 rounds.

Figure 6 shows the distribution of group leaders to the depth of the tree. In our simulation, the height of the tree is 44. As discussed in 3.2, the root has a depth of 0, and nodes in depth 44 are all leaves. Since these nodes only have the count of 1, the chances of being leaders are almost 0. Due to the small counts, nodes with depth of 40 or more have no chance of being leaders, either. On the other hand, nodes with depth between 10 and 30 have higher probabilities to become leaders. For example, an average of 5.5 nodes in the depth of 22 are group leaders. The root is always a leader, so the average number of group leader in depth 0 is 1.

Figure 7 shows the distribution of group sizes when there is no attack. As can be seen, the mean of the group size is 30, and the

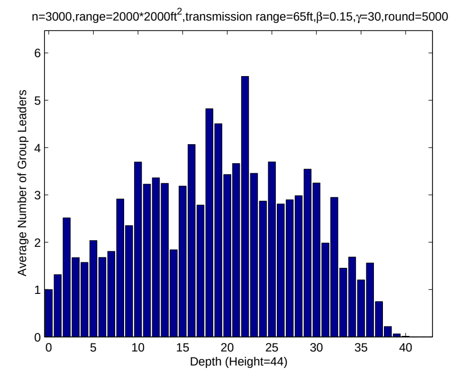

Figure 6: The distribution of group leaders

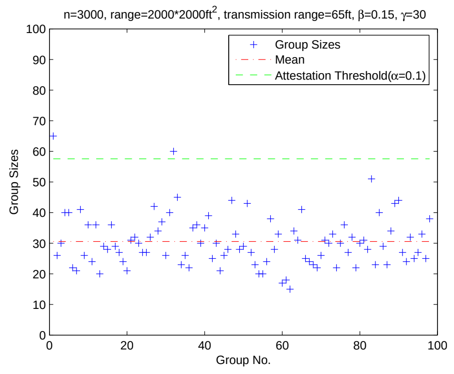

Figure 7: The distribution of group sizes

resulted group sizes do not deviate much from the mean. More specifically, most group sizes are limited between 20 and 40. This can provide a good basis for the attestation. According to the critical value in Grubbs' test, when the total group number is 98 and $\alpha = 0.1$ (one tailed test, $\alpha/2 = 0.05$), the attestation threshold is 57.57. If the attacker increases the group size larger than this threshold, the BS can detect this attack by choosing the corresponding group for attestation. From the figure, we can see that the number of attestations are very small when there is no attack. There are only 2 out of 98 legitimate groups with group sizes larger than this threshold. Although the BS will also choose these two groups for attestation, the BS will accept their values after the attestation because they are both legitimate groups.

5.3 Communication Overhead

In this section, we first analyze the communication overhead of our protocol and then further use simulations to verify our conclusion that our protocol only causes little extra overhead compared to hop-by-hop aggregation. To accurately measure the overhead, we use the metric of $packet * hop$ and $byte * hop$ (product of the data size and the message traveling distance), because message overhead is proportional to the traveling distance of sensing data. To help understand the communication overhead of our protocol, we also compare it with the no-aggregation and hop-by-hop aggregation approaches. For ease of exposition, we do not consider the impact of packet retransmission due to the unreliable channel. Although packet retransmission will increase the absolute performance overhead of SDAP, we expect its *relative* performance overhead compared to the other two approaches will be similar because packet retransmissions also occur in these approaches. We will verify the above intuition quantitatively in our future work.

The following notations are used in the analysis:

- (n, d, h) is used to model the aggregation tree, where n means the total number of nodes, d is the degree of the tree (e.g., d=2 represents a binary tree), and h is the height of the tree;
- n_g is the number of attested groups;
- n_p is the number of attestation paths in the attested group (for ease of expression, we assume the number of attestation path in each attested group is the same);
- $g(1 \leq g \leq n)$ is the average group size.

5.3.1 *Analytical Results in* $packet * hop$

Since all the three schemes have the query broadcast overhead, we only compare the communication overhead in the aggregation (including attestation for our protocol). In the hop-by-hop data aggregation approach, the number of packets is equal to the number of edges in the broadcast tree. Hence, the communication overhead of the hop-by-hop aggregation approach is as follows:

$$C_{hop-by-hop} = n - 1 = \Theta(n).$$

On the other hand, without in-network aggregation, i.e., every sensor node sends its reading (with a MAC) separately to the BS, the communication overhead can be expressed by:

$$\begin{aligned} C_{no-aggregation} &= \textstyle\sum_1^h i \cdot d^i \\ &= \frac{hd^{h+2} - (h+1)d^{h+1} + d}{(d-1)^2} \\ &= \Theta(n \cdot logn), \end{aligned}$$

because h can be approximated by $logn$. The upper bound is $O(n^2)$ in case of a linear tree ($h = n$, $d = 1$).

In our protocol, the total number of groups is $\lfloor n/g \rfloor + 1$, considering the extra group with the BS as the default leader. The height of the group can be approximated by $\lceil h/2 \rceil$, and then the average distance to the BS from a leader is $\lfloor h/2 \rfloor$. Based on the results shown in the Figure 6, this assumption is reasonable because we only consider the average case. With these assumptions, the communication overhead of our protocol during the aggregation phase is $(g-1)(\lfloor n/g \rfloor + 1) + \lfloor n/g \rfloor \lfloor h/2 \rfloor$.

The overhead for attestation depends on the number of attested groups and the attestation paths that we have chosen. The overhead of disseminating the attestation request is $n_g n_p \lfloor h/2 \rfloor$, and the overhead of sending the data back to the BS is $n_g n_p [\lfloor h/2 \rfloor + \sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i)d]$. Therefore, the total overhead is:

$$\begin{aligned} C_{our} &\leq (g-1)(\lfloor n/g \rfloor + 1) + \lfloor n/g \rfloor \lfloor h/2 \rfloor + n_g n_p \lfloor h/2 \rfloor \\ &\quad + n_g n_p [\lfloor h/2 \rfloor + \textstyle\sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i)d] \\ &\approx n + \lfloor nh/2g \rfloor + n_g n_p h + \frac{n_g n_p dh(3h+2)}{8}. \end{aligned}$$

This formula actually gives us an upper bound of the communication overhead because in case of multiple attestation paths, a node locating on multiple paths only needs to report one copy of its aggregate. Also, the n_p attestation requests for a group is actually piggybacked into one packet.

The communication overhead of our protocol depends on the average group size g. If g is as large as n, the overhead is about $O(n)$. Otherwise, if g is very small and can be treated as a constant number, the overhead is $O(n \cdot logn)$. In either case, the overhead of our protocol is less than the no-aggregation approach and higher than the hop-by-hop aggregation approach.

To quantify the difference, data results in $packet * hop$ can be seen in the Figure 8. The results are based on the following parameter setup: $n = 3280$, $d = 3$, $h = 7$ and $n_p = 1$. As shown in the figure, the communication overhead of our protocol is between $3.4K$ and $4.4K$. Using the same parameters, we can easily calculate the cost of the other approaches. Specifically, the communication overhead of the hop-by-hop aggregation approach is $3K$,

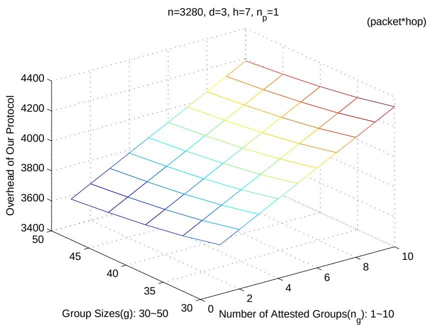

Figure 8: Overhead in packet*hop

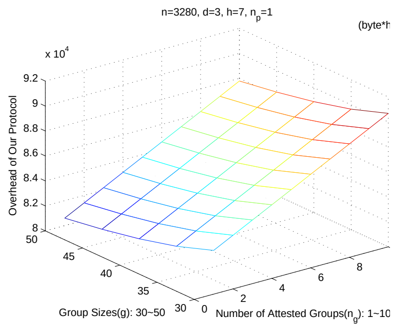

Figure 9: Overhead in byte*hop

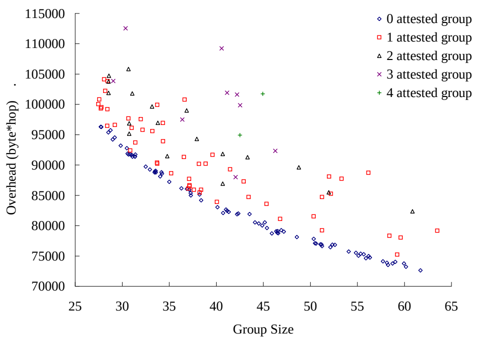

Figure 10: Overhead based on simulations

and the communication overhead of the no-aggregation approach is $21K$. Thus, our protocol does not add much overhead compared to the hop-by-hop aggregation.

5.3.2 *Analytical Results in* $byte * hop$

With the results of last subsection, we can easily calculate the overhead in $byte * hop$. Each packet includes node id (2 bytes), data (4 bytes) and MAC (8 bytes), so the overhead of the hop-by-hop aggregation and the no-aggregation approaches in $byte * hop$ are the results in $packet * hop$ multiplied by 14(bytes).

Although we did not consider the query dissemination overhead, for fair comparison, we should consider the extra overhead of our protocol, due to the 4-byte random number used in the query dissemination. The total extra communication overhead for the query broadcast from the BS is about $4(n-1)$. The committed aggregation packet is of the same format and with a size of 19 bytes (2 bytes for id, 5 bytes for data including counts, 8 bytes for MAC and 4 bytes for the grouping seed), thus the overhead of the aggregation is $19[(g-1)(\lfloor n/g \rfloor + 1) + \lfloor n/g \rfloor \lfloor h/2 \rfloor]$. The size of the attestation request from the BS is 10 bytes, 2 bytes for the leader id and 8 bytes for grouping/attestation seeds. Thus, the overhead to disseminate the attestation request is $10 n_g n_p \lfloor h/2 \rfloor$. The overhead of sending data back to the BS is $17 n_g n_p [\lfloor h/2 \rfloor + \sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i)] + 9 n_g n_p \sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i)(d-1)$, because the packets sent back to the BS from the nodes on the attestation path have the size of 9 bytes (2 bytes for id and 7 bytes for data) and packets from other nodes in the attestation are of the size 17 bytes (2 bytes for id and 15 bytes for data). The total communication overhead of our protocol in $byte * hop$ is given by

$$\begin{aligned} C'_{our} &\leq 4(n-1) + 19[(g-1)(\lfloor n/g \rfloor + 1) + \lfloor n/g \rfloor \lfloor h/2 \rfloor] \\ &\quad + 27 n_g n_p \lfloor h/2 \rfloor + 17 n_g n_p \textstyle\sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i) \\ &\quad + 9 n_g n_p \textstyle\sum_1^{\lceil h/2 \rceil} (\lfloor h/2 \rfloor + i)(d-1) \\ &\approx 23n + \lfloor \frac{19nh}{2g} \rfloor + \lfloor \frac{27 n_g n_p h}{2} \rfloor + \frac{n_g n_p h(3h+2)(9d+8)}{8}. \end{aligned}$$

The result in $byte*hop$ (Figure 9) is based on the same parameter setup: $n = 3280$, $d = 3$, $h = 7$ and $n_p = 1$. As shown in the figure, the communication overhead of our protocol is between $80K$ and $92K$, whereas the communication overhead of the hop-by-hop aggregation approach is $45.9K$ and the communication overhead of the no-aggregation approach is $298.5K$.

5.3.3 *Simulation Results*

The previous analytical results are applicable to balanced trees with fixed degrees. To evaluate the communication overhead for more general cases, we setup a simulation testbed. In our simulation, 3000 nodes are randomly distributed in an area of $2000 * 2000 ft^2$. The transmission range is set to $60ft$. To test different group sizes, (β, γ) takes 8 different values: $(0.15, 30)$, $(0.14, 33)$, $(0.13, 36)$, $(0.12, 39)$, $(0.11, 42)$, $(0.10, 45)$, $(0.09, 48)$, $(0.08, 51)$. For each pair of parameters, we run the simulation 20 times, each time with a different grouping seed. Based on our Grubbs' test, among all the 160 simulation runs, there are no attested groups in 79 simulations, 1 attested group in 52 simulations, 2 attested groups in 18 simulations, 3 attested groups in 9 simulations, and 4 attested groups in 2 simulations. We choose one attestation path in each attested group.

The simulation result in $byte * hop$ is shown in Figure 10. As can be seen from the figure, the overhead of our protocol including attestation is between 70K$\sim$115K. With the same parameters, through simulation, we get the communication overhead of the hop-by-hop aggregation approach to be $42K$, and the communication overhead of the no-aggregation approach to be $1202K$.

In summary, through analytical and simulation results, we can see that our protocol does not add much overhead compared to the hop-by-hop aggregation approach, but is more secure. On the other hand, as the no-aggregation approach, our protocol provides security, but with much less communication overhead.

6. RELATED WORK

Many data aggregation protocols [2, 3, 4, 5, 6, 8, 21] have been proposed, but none of them were designed with security in mind. Until recently very few work has been focused on secure data aggregation.

After analyzing the possible attacks on the existing aggregation primitives, Wagner[9] proposed a mathematical framework for formally evaluating the security of several resilient aggregation techniques. For example, median is a more robust estimator than mean; truncation and trimming can be used to eliminate possible outliers. This work, however, is not really about data aggregation because it assumes the BS has already collected all the raw data. Also, abnormal data are discarded without further reasoning. Hu and Evans [11] propose a secure hop-by-hop data aggregation scheme that works if one node is compromised. They also assume that only leaf nodes in the tree topology sense data whereas the intermediate nodes do not have their own readings. SDAP can tolerate more compromised nodes and allows every node to input its own readings.

Du et al. [22] proposes a mechanism that allows the base station to check the aggregated values submitted by several designated aggregators, based on the endorsements provided by a certain number of witness nodes around the aggregators. Their scheme does not provide per-hop aggregation. Also it is assumed that sensing nodes can be trusted and witness nodes do not collude with the aggregators. However, this condition may not always hold in practice.

Przydatek et al. [23] present SIA, a Secure Information Aggregation scheme for sensor networks where a fraction of sensor nodes may be compromised. In their model, the BS is the only aggregator, which collects the authenticated raw data from all the sensor nodes in the network. The aggregator then computes an aggregation result over the raw data together with a commitment to the data based on a Merkle-hash tree and then sends them to a trustable remote user, who later challenges the aggregator to verify the aggregate. They assume that the bandwidth between a remote user and an aggregator is a bottleneck; therefore, their protocol is for reducing this bandwidth overhead while providing a means to detect with high probability if the aggregator is compromised. The main difference between SIA and SDAP is that SIA does not deal with per-hop aggregation because it assumes the raw data are first collected by the aggregator. Since SIA and SDAP work in different stages with different network models (e.g., in SDAP there is no remote user), in our future work we will investigate the potential of integrating these two.

Several other works [24, 25, 26] had also proposed various solutions to prevent false data injection attacks in sensor networks. In their model, it is assumed that a set of sensors are deployed as a cluster in an area of interest. When these sensors reach an agreement on an event, each of them will contribute a MAC over the event report. If a forwarding node shares a MAC key with the endorsing sensors, it will be able to verify the authenticity of the report. It drops the report if the verification fails. In this way, an injected false data packet could be discarded before it reaches the BS, saving the forwarding energy. We note that although these schemes also address the problem of false data injection, they do not involve data aggregations.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we propose SDAP, a Secure Data Aggregation Protocol for large-scale sensor networks. By using *divide-and-conquer*, we partition the aggregation tree into groups to reduce the importance of high-level nodes in the aggregation tree; we use *commit-and-attest* so that the BS has a way to verify the aggregates.

In the future, we will further enrich the protocol in more detail. For example, the breadth-based attestation and the content-based attestation techniques may also be included in the protocol. We may implement and show the benefits of Bloom Filter in our protocol. The potential of integrating with different network model, such as that in SIA, will be investigated too.

Acknowledgement: We would like to thank our shepherd Srikanth Krishnamurthy for his suggestions on improving this paper. We also thank the anonymous reviewers for their comments and suggestions.

8. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of ACM Mobicom*, Seattle, Washington, USA, August 1999, pp. 263–270, ACM.

[3] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *ICDCS*, 2002, pp. 457–458.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *MOBICOM*, 2000, pp. 56–67.

[5] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *International Workshop on Distributed Event-Based Systems, (DEBS '02)*, Vienna, Austria, July 2002.

[6] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "TAG: A tiny aggregation service for ad-hoc sensor networks," in *OSDI*, 2002.

[7] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Mobile and Ubiquitous Systems: Networking and Services MobiQuitous 2005*, July 2005.

[8] Jen-Yeu Chen, Gopal Pandurangan, and Dongyan Xu, "Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis," in *IPSN*, 2005, pp. 348–355.

[9] David Wagner, "Resilient aggregation in sensor networks," in *Proceedings of ACM Workshop SASN '04*, 2004.

[10] "Mica Motes," `http://www.xbow.com`.

[11] L. Hu and David Evans, "Secure aggregation for wireless networks," in *Workshop on Security and Assurance in Ad hoc Networks*, January 2003.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proc. of ASPLOS IX*, 2000.

[13] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: security protocols for sensor netowrks," in *Mobile Computing and Networking*, 2001, pp. 189–199.

[14] Donggang Liu and Peng Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of ACM CCS*, October 2003.

[15] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of ACM CCS*, October 2003.

[16] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least Privilege and Privilege Deprivation: Towards Tolerating Mobile Sink Compromises in Wireless Sensor Networks," *ACM MobiHoc*, May 2005.

[17] J. McCune, E. Shi, A. Perrig, and M. Reiter, "Detection of denial-of-message attacks on sensor network broadcasts," in *IEEE Symposium on Security and Privacy*, 2005, pp. 64–78.

[18] Ralph Merkle, "A certified digital signature," in *Proceedings of Advances in Crypto-89*, 1989, pp. 218–238.

[19] Burton H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] Grubbs Frank, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, February 1969.

[21] Yong Yao and Johannes Gehrke, "The Cougar approach to in-network query processing in sensor networks," *SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

[22] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A witness-based approach for data fusion assurance in wireless sensor networks," in *Proc. of IEEE GLOBECOM '03*, December 2003.

[23] B. Przydatek, D. Song, and A. Perrig, "SIA: secure information aggregation in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 255–265.

[24] Fan Ye, Haiyun Luo, Songwu Lu, and Lixia Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proceedings of IEEE Infocom'04*, 2004.

[25] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," *IEEE INFOCOM*, March 2005.

[26] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proceedings of IEEE Symp. on Security and Privacy*, 2004, pp. 259–271.