Secure Data Aggregation without Persistent Cryptographic Operations in Wireless Sensor Networks

Kui Wu*, Dennis Dreef
Computer Science Dept.
University of Victoria
BC, Canada V8W 3P6
{wkui,ddreef}@uvic.ca

Bo Sun
Computer Science Dept.
Lamar University
Beaumont, TX 77710, USA
bsun@cs.lamar.edu

Yang Xiao
Computer Science Dept.
University of Memphis
Memphis, TN 38152, USA
yangxiao@ieee.org

Abstract

In-network data aggregation is an essential operation to reduce energy consumption in large-scale wireless sensor networks. With data aggregation, however, raw data items are invisible to the base station and thus the authenticity of the aggregated data is hard to guarantee. A compromised sensor node may forge an aggregation value and mislead the base station into trusting a false reading. Due to the stringent constraints of energy supply and computing capability on sensor nodes, it is challenging to detect a compromised sensor node and keep it from cheating. This paper proposes a Secure Aggregation Tree (SAT) to detect and prevent cheating. Our method is essentially different from other existing solutions in that it does not require any cryptographic operations when all sensor nodes work honestly. The detection of cheating is based on the topological constraints in the aggregation tree.

1 Introduction

Extremely small sensors have been used broadly for various applications such as habitat monitoring, battlefield surveillance, and forest fire monitoring. A large number of tiny sensors collect measurement data and rely on multihop short-range radio communication to send data to the processing center, which is also called the base station or the sink node. The lifetime of sensors depends on effective energy saving strategies such as sensor scheduling and in-network information processing to reduce the data traffic to the base station. One important type of in-network processing is data aggregation.

While data aggregation can effectively reduce the amount of data transmitted to the base station, raw data items may be invisible to the base station and thus their authenticity and integrity are hard to guarantee. As such, data aggregation is potentially vulnerable to attackers who may inject bogus information or forge aggregated values without being detected. For instance, if a sensor close to the base station is compromised, it may claim a fake aggregation value to the base station and mislead the base station into trusting the fake information. It may have a disastrous impact if end users respond according to the faulty information.

Some methods have been proposed to solve the above problem [2, 3, 7, 9, 10]. Existing methods depend on complex data authentication operations [2, 3, 10] or the statistical features of specific aggregation operations such as the mean, max, and min [1, 8, 9]. To guarantee correctness, *persistent* authentication operations are used in most existing methods. *Persistent* authentication, though very safe, may consume too much energy and is thus undesirable for sensor networks. Besides the energy concern, another barrier of using complex cryptographic operations is that the calculation capability of sensor nodes is very limited. For this reason, cryptographic operations may significantly increase the data processing delay and renders the network performance intolerable.

In this paper, we assume the existence of a secure mechanism but use it only when necessary, i.e., only when cheating activities are detected. This strategy is fundamentally different from existing solutions in that its security is based on cheating detection instead of persistent data authentication. This strategy can reduce energy consumption and more importantly can save the CPU resource. It, however, requires the support of a lightweight and scalable cheating detection method, which is a quite challenging task.

This paper is motivated to solve this problem. First, it introduces some topological constraints when

*This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Foundation for Innovation (CFI).

building a Secure Aggregation Tree (SAT), which facilitates the monitoring of the behavior of each aggregation sensor node. Second, when the aggregated values from an aggregation node are in doubt, a weighted voting scheme is proposed to decide finally whether the aggregation node is properly behaving or is cheating. Third, if a misbehaving node is detected, a local recovery scheme is presented to re-build SAT so that the misbehaving node is excluded from the aggregation tree. Since no cryptographic operations are required when all nodes work honestly, our method is lightweight. Since no centralized operations are needed at the base station, our method also scales very well.

2 Assumptions

We assume that a node cannot impersonate its neighbors. This is not a strong assumption since if a node, A, is in the promiscuous listening mode, it can quickly detect any transmission from its neighboring nodes with A's IP address. This assumption is to exclude the possibility that a node impersonates its father node in the aggregation tree to send false aggregated data such that an honest father is voted out of the network.

We assume that two neighboring sensor nodes have mechanisms for secure communication so that they can decrypt and authenticate each other's messages. There are a lot of key distribution methods proposed for sensor networks that can meet this requirement [4, 5, 6]. This seemly strong assumption does not mean that our method will always use message encryption and authentication. In contrast, they are used *only when cheating behavior is detected.* If all sensor nodes work honestly, no cryptographic operations are required.

In this paper, we focus only on the integrity and authenticity of data.

3 Building a Secure Aggregation Tree

3.1 The structure of SAT

If we could build an aggregation tree such that any child node can monitor the behavior of its father node, then the cheating activities of any non-leaf (aggregation) nodes can be detected. In this paper we do not consider the cheating behavior of leaf nodes since it is indistinguishable between cheating and malfunctioning for a leaf node. Obviously, to allow a child node to monitor its father node's behavior, it is required that the child node should be able to know all the messages from its sibling nodes to the father node. In other words, a father node together with its children nodes should form a *clique*.

3.2 A Distributed Algorithm to build SAT

We propose a distributed algorithm, which builds the aggregation tree starting from the sink node and includes four steps as follows.

1. Step 1: The sink node locally broadcasts an *invitation* message to all of its one-hop neighbors, indicating that they should be its children. The *invitation* message includes the IDs of all nodes that a father node wants to invite to join the aggregation tree as its children. It should also include the hop count value to make a node aware of its minimal hop count to the sink node. The hop count value in the *invitation* message from the sink node is set to zero. A node sets its initial hop count value to infinity and updates its hop count value as one plus the minimal hop count value in all received *invitation* messages.

2. Step 2: Once a node receives an *invitation* message, if this node has not joined the aggregation tree and the *invitation* message includes this node as a child node, then this node joins the aggregation tree and records the sender of the *invitation* message as its father node. It locally broadcasts a *join* message to notify its neighbors about this decision. This *invitation* message is also called *activating invitation* message since it enforces the node to join the aggregation tree. Once a node joins the tree, later received *invitation* messages will be recorded for future use if the hop count value in the *invitation* messages is smaller than the node's current hop count value. More specifically, these *invitation* messages will be used to select a candidate aggregation node if a current working aggregation node is compromised, as described in detail in Section 4.3. To avoid forming a poor aggregation tree, an additional rule should be applied, although this rule is rarely used if the network density is high: if a node receives an invitation message but the hop count value included in the message is 2 hops larger than its current hop count value, then this invitation message is ignored. We call this rule the *optimization rule* since it excludes the possibility of creating a poor aggregation tree that requires large energy consumption on data delivery.

3. Step 3: After a node joins the aggregation tree, by checking its one-hop and two-hop neighbors, excluding those indicated in the *activating invitation* message, it finds all the cliques that it belongs to. If such cliques cannot be found, then

this node works as a leaf node. Otherwise, it selects the maximal clique and locally broadcasts an *invitation* message with the hop count value increased by one, indicating that all other nodes in the selected clique should be its children.

4. Step 4: Repeat step 2 and step 3 until all non-isolated nodes have joined the tree.

Due to the topological constraint that an aggregation node together with its children should form a clique, it is possible that some nodes may not join the aggregation tree even if they have paths to the sink node. We call such nodes *sparse nodes* since they have only sparse set of neighboring nodes. Nevertheless, as demonstrated later, the ratio of the number of sparse nodes over the total number of sensor nodes is *extremely small* if the network density is reasonably high. As such, we could simply require the sparse nodes to send their messages to the sink node without performing any in-network processing. Any secure routing and secure unicast mechanisms could be applied to the messages sent from the sparse nodes.

At the end of the SAT buildup process, each node should record the following information:

1. the IDs of its one-hop and two-hop neighboring nodes,
2. the ID of its father node in the aggregation tree,
3. the IDs of its sibling nodes in the aggregation tree,

4 Cheating Detection for Data Aggregation

4.1 Detection of Potential Misbehavior

Due to the topological constraints in SAT, each node can hear all messages sent to its father node and can monitor the message sent from its father node to its grandfather node to check if the father node performs data aggregation correctly. If a node's father node sends out a value that is *significantly* different from a correct aggregation value, the node will raise an *alert*. The criterion of setting a proper threshold value for raising an alert will be discussed in Section 5. In this paper, we differentiate an *alert* message from a *detection-confirmation* message. An *alert* message means a potential compromise, while a *detection-confirmation* message indicates the final confirmation of the existence of a cheating node.

To draw a consistent conclusion about an aggregation node's behavior, we assign each *alert* with a confidence value. This confidence value is calculated based on the specific aggregation function and is used to indicate the likelihood that the aggregation node is cheating. In the next section, we propose a weighted voting method to make the final decision regarding whether or not an aggregation node is cheating.

4.2 Weighted Voting

First of all, we want to remark that all control messages in the weighted voting process and the local recovery process should be encrypted and authenticated with some underlying secure communication mechanisms. This is to guarantee the correct final decision once potential misbehavior is detected. Also, we stress again that *the secure communication is required only when an aggregation node is working improperly.*

Once a sensor node detects that its father node might be cheating, it sends out an *alert* message to all its neighbors except the father node. To keep a compromised node from sending out fake alert messages and also to keep the voting process invisible to the father node, the alert message should be encrypted and authenticated. Note that the voting process should be secure to avoid potential fake control messages.

An *alert* message should include the following information:

1. The cheating node's ID,
2. The detecting node's ID,
3. The confidence value of the alert.

The confidence value indicates the likelihood of a correct detection. A large confidence value indicates that the aggregation node is more likely to be cheating.

Once receiving *alert* messages from neighboring nodes, a node first checks if the cheating node is its father node. If yes, it calculates the weighted confidence value using the following formula:

$$F = \frac{\sum_{i=1}^{m_1} f_i}{m}$$

where f_i is the confidence value included in the alert message from the sibling node i, m is the total number of sibling nodes, and m_1 is the number of sibling nodes that send out an *alert* message.

If the weighted confidence value F is larger than a predefined threshold value, the father node is assumed to be cheating, and a *detection-confirmation* message will be broadcast within a given hop limit. Any node receiving the *detection-confirmation* message will use the following local recovery mechanism to avoid using the compromised node.

4.3 Local Recovery

If a sensor node, A, receives a *detection-confirmation* message, it checks if the cheating node is one of its child nodes or its father. If the cheating node is one of its child nodes, A will ignore any messages from that child.

If the cheating node is its father node, A will select another candidate father from its recorded *invitation* messages. Note that deadlocks cannot be formed since a node only records *invitation* messages that have smaller hop count values than its current hop count value. In other words, sensor node A selects another neighboring sensor node, B, as its father node *only if* B has a smaller hop count value to the sink node than A. This is to avoid the deadlock situation where two sibling nodes select each other as the potential father node. If a candidate father node cannot be found, then the current node becomes an isolated sensor node since none of its fathers can be used for reliable aggregation.

The newly formed "isolated" sensor nodes actually have a physical path to the sink node, although the path has to include some misbehaving nodes and may not be secure. If the sensor network still requires the sensing data from these "isolated" sensor nodes, more secure mechanisms such as data signature and authentication must be utilized for each message from the newly formed "isolated" nodes.

5 The Criterion of Raising alerts

5.1 Problem Modeling

There are mainly two reasons that the aggregated value at a father node is different from that calculated at a child node. The first is that some packets sent to the father node may be lost, and the other is that the father node uses a slightly different set of values for aggregation due to time asynchrony. In our model, a father node knows the number of its child nodes. When an aggregation calculation is required, the father node will use the most recently received values from its child nodes as the input.

For instance, as shown in Figure 1, the child nodes, $C_1, C_2, \ldots, C_m$, send data packets to the father node, F. Assume that during a time period each child node $C_i (i = 1, \ldots, m)$ sends n packets to the father node, denoted as $D(C_i, 1), D(C_i, 2), \ldots, D(C_i, n)$ respectively. Due to packet losses or time asynchrony, the value of the aggregation function calculated by the father node might be any value in the set $\{f(D(C_1, j_1), D(C_2, j_2), D(C_3, j_3), \ldots, D(C_m, j_m))\}$ where f is the aggregation function and $j_k, k = 1, 2, \ldots, m$ might be any value in $\{1, 2, \ldots, n\}$.

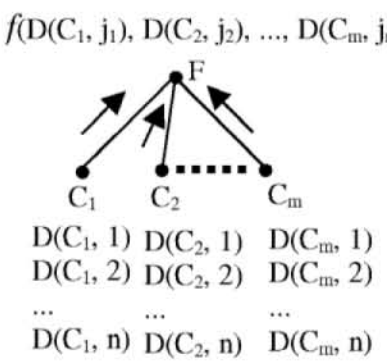

Figure 1: An illustration of the data aggregation model

For a child node, when it monitors its father's behavior, it calculates the same aggregation function

$$f(\overline{D(C_1, j_1)}, \overline{D(C_2, j_2)}, \overline{D(C_3, j_3)}, \ldots, \overline{D(C_m, j_m)}))$$

where $\overline{D(C_i, j_k)} (i = 1, \ldots, m)$ is the newest data received from the sibling node i. As a reasonable assumption, we assume that $|D(C_i, j_k) - \overline{D(C_i, j_k)}|$ is bounded by δ, since it is unlikely that during a short time period readings from the same sensor change dramatically. We assume that data from different child nodes are independent.

Our problem could then be modeled as follows. Given an aggregation function f, an arbitrary positive value λ, a set of values $D_1, D_2, \ldots, D_m$, and another set of values $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ with each $\overline{D_i}$ randomly selected from $[D_i - \delta, D_i + \delta] (i = 1, 2, \ldots, m)$ respectively, what is the probability that $|f(D_1, D_2, \ldots, D_m) - f(\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m})| \geq \lambda$)? We denote the above probability as P_c.

The value of λ indicates the absolute difference between the aggregated value calculated at a father node and that monitored at a child node. Given a small threshold value θ, if $P_c \leq \theta$, then the child node should raise an alert since it is unlikely (i.e., P_c is too small) that the father node should generate such a different aggregated value. The confidence value for the alert is set to $1 - P_c$.

5.2 Mean

Denote the mean of $D_1, D_2, \ldots, D_m$ as $S = \frac{\sum_{i=1}^{m} D_i}{m}$. Also denote the mean of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{S} = \frac{\sum_{i=1}^{m} \overline{D_i}}{m}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that S is a constant and $\overline{S}$ is a random variable. We are asked to estimate $P_c = Pr(|\overline{S} - S| \geq \lambda)$. By calculation, we get

$$P_c \begin{cases} = 0 & \text{if } \lambda > S + \delta \text{ or } \lambda < S - \delta \\ \leq \frac{\delta^2}{3\lambda^2} & \text{otherwise} \end{cases}$$

5.3 Min

Denote the *min* of $D_1, D_2, \ldots, D_m$ as $m = min\{D_1, D_2, \ldots, D_m\}$. Also denote the *min* of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{m} = min\{\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}\}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that m is a constant and $\overline{m}$ is a random variable. We are asked to estimate $P_c = Pr(|\overline{m} - m| \geq \lambda)$. By calculation, we get

$$P_c = \begin{cases} 0 & \lambda > \delta \\ 1 + \prod_{i=1}^{m} \frac{\delta + D_i - m - \lambda}{2\delta} - \prod_{i=1}^{m} \frac{\delta + D_i - m + \lambda}{2\delta} & \text{else} \end{cases}$$

5.4 Max

Denote the *max* of $D_1, D_2, \ldots, D_m$ as $M = max\{D_1, D_2, \ldots, D_m\}$. Also denote the *max* of $\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}$ as $\overline{M} = max\{\overline{D_1}, \overline{D_2}, \ldots, \overline{D_m}\}$, where $\overline{D_i}$ is randomly selected from $[D_i - \delta, D_i + \delta]$ respectively. Note that M is a constant and $\overline{M}$ is a random variable. The calculation of $P_c = Pr(|\overline{M} - M| \geq \lambda)$ is the same as that when the aggregation function is *min* in the previous section. To save space, we omit its result.

6 Simulation Study

6.1 Simulation Model

In this section, we perform simulation study to further demonstrate the feasibility and the effectiveness of SAT. We want to illustrate that SAT does not introduce extra transmission cost compared to other commonly used tree structures and the proportion of the sparse nodes in SAT is extremely small for dense networks. In addition, we want to justify that the cost for local recovery is practically acceptable.

For comparison purposes, we implemented SAT and other types of tree structures, namely, BFT (Breadth-First Tree) and BFT-D (Breadth-First Tree with the Distance constraint). In BFT-D, we list node A as node B's child during the breadth-first search only when node A is further away from the sink than node B. The simulation was run on a variety of scenarios. The sensor nodes were deployed randomly in a 1000 meters by 1000 meters area with nodes having a transmission range of 50 meters. We changed the number of sensor nodes from 500 to 5000 to change network density and placed the sink node in four different locations, i.e., the up-left corner $(0, 0)$, the top-middle $(500, 0)$, the left-middle $(0, 500)$, and the center $(500, 500)$. We ran each scenario with 3 different random seeds. The simulation results were obtained by calculating the average of all runs.

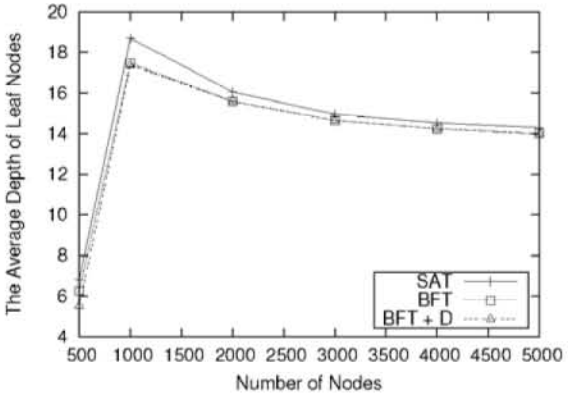

Figure 2: Comparison of Different Tree Structures

6.2 Simulation Results

6.2.1 The average depth of leaf nodes

Figure 2 shows results of the average depth of the leaf nodes with different tree structures. From the results, the three tree structures, SAT, BFT, and BFT-D, have very close performance. The average depth of leaf nodes indicates the average height of the aggregation tree and thus the approximate cost on message delivery with the aggregation tree. Figure 2 illustrates that SAT, despite its advantage for aggregation monitoring, does not incur obvious extra overhead on message delivery compared to other types of trees.

6.2.2 The ratio of sparse nodes

Figure 3 shows the ratio of the number of sparse nodes over the total number of nodes. From the figure, we can see that when the network is sparse, e.g., when the number of nodes is smaller than 1000, the number of sparse nodes is not negligible. Nevertheless, when network density becomes higher, the performance of SAT becomes significantly better with the ratio of sparse nodes quickly dropping to almost zero. This phenomenon justifies our previous claim that with dense networks, the cost on message delivery from the sparse nodes is negligible.

6.2.3 The number of local candidate paths

Table 1 shows the number of candidate paths from a node to its grandfather node without using its current father node on the SAT. Such candidate paths could be used for local recovery once an aggregation node is detected to behave abnormally. We can see that with a looser constraint on the hop count from a node to its grandparent, more candidate paths could

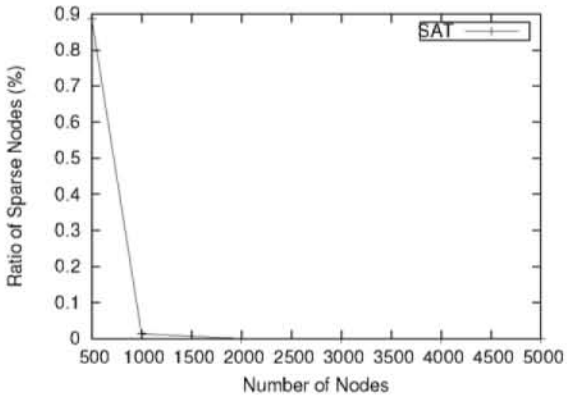

Figure 3: The Ratio of sparse nodes

Table 1: Number of Candidate Paths

Nodes	1 Hop	2 Hops	3 Hops	4 Hops
500	0.058997	0.572271	3.277286	18.79056
1000	0.022493	0.761523	7.954369	77.62297
2000	0.003250	0.889625	23.00842	444.9818
3000	0.002028	0.933139	45.53275	1323.001
4000	0.001917	0.937104	75.66433	2975.243
5000	0.001933	0.960550	113.3157	5620.514

be found and thus the chance of successful local recovery is higher. But a candidate path with larger hop count requires more message transmissions. There is a tradeoff between the possibility of local recovery and the recovery cost. We observe that with a hop count constraint of 3, more than 15 local candidate paths could be found in most cases.

7 Conclusion

Traditional solutions to secure data aggregation use persistent data authentication that incurs heavy computational overheads even if no attackers or misbehaving nodes exist in the network. Such computational overheads waste energy and may greatly reduce the available CPU resource for data processing. An ideal security solution should not perform any cryptographic operations when the network works correctly, and uses data authentication only when misbehaving nodes are detected. For this purpose, we propose a Secure Aggregation Tree (SAT) with which it is very easy to observe the behavior of aggregation nodes without resorting to persistent data authentication. With analysis and simulation, we demonstrate the feasibility and effectiveness of using SAT to monitor the aggregation operations.

References

[1] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: an energy - accuracy tradeoff," *Elsevier Ad-hoc Networks Journal (special issue on sensor network protocols and applications)*, 2003.

[2] H. Cam, S. Ozdemir, P. Nair, and D. Muthuavinashiappan, "Espda: Energy-efficient and secure pattern-based data aggregation for wireless sensor networks," in *IEEE Sensors 2003 Conference*, Toronto, Canada, October 22-24 2003.

[3] L. Hu and D. Evans, "Secure aggregation for wireless networks," 2003.

[4] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, November 2004.

[5] A. Perrig, "The biba one-time signature and broadcast authentication protocol," in *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, Philadelphia PA, USA, Nov. 2001, pp. 28–37.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "Spins: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sept. 2002.

[7] B. Przydatek, D. Song, and A. Perrig, "Sia: Secure information aggregation in sensor networks," in *ACM SenSys 2003*, Nov 2003.

[8] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems.* New York, NY, USA: ACM Press, 2004, pp. 239–249.

[9] D. Wagner, "Resilient aggregation in sensor networks," in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, October 2004.

[10] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration based approach," in *IEEE INFOCOM*, March 2005.