Title	Efficient and Optimally Secure In-Network Aggregation in Wireless Sensor Networks
Author(s)	Miyaji, Atsuko; Omote, Kazumasa
Citation	Lecture Notes in Computer Science, 6513/2011: 135-149
Issue Date	2011
Type	Journal Article
Text version	author
URL	http://hdl.handle.net/10119/9846
Rights	This is the author-created version of Springer, Atsuko Miyaji, Kazumasa Omote, Lecture Notes in Computer Science, 6513/2011, 2011, 135-149. The original publication is available at www.springerlink.com, http://dx.doi.org/10.1007/978-3-642-17955-6
Description	The 11th International Workshop on Information Security Applications. WISA 2010, Jeju Island, Korea, August 24-26, 2010.

Efficient and Optimally Secure In-Network Aggregation in Wireless Sensor Networks

Atsuko Miyaji and Kazumasa Omote

Japan Advanced Institute of Science and Technology (JAIST)
{miyaji,omote}@jaist.ac.jp

Abstract. In many wireless sensor network applications, the data collection sink (base station) needs to find the aggregated statistics of the network. Readings from sensor nodes are aggregated at intermediate nodes to reduce the communication cost. However, the previous optimally secure in-network aggregation protocols against multiple corrupted nodes require two round-trip communications between each node and the base station, including the *result-checking phase* whose *congestion* is $\mathcal{O}(\log n)$ where n is the total number of sensor nodes.
In this paper[1], we propose an efficient and optimally secure sensor network aggregation protocol against multiple corrupted nodes by a *weak adversary*. Our protocol achieves one round-trip communication to satisfy optimal security without the result-checking phase, by conducting aggregation along with the verification, based on the idea of TESLA technique. Furthermore, we show that the congestion is constant. This means that our protocol suits large-scale wireless sensor networks.

1 Introduction

In many wireless sensor network applications, the data collection sink (base station) needs to find the aggregate statistics of the network. Readings from sensor nodes are aggregated at intermediate nodes to reduce the communication cost. This process is called in-network aggregation [1–6, 18–20]. Since aggregation reduces the amount of data to be transmitted through the network, it consequently decreases bandwidth consumption and energy depletion.

Security is a critical requirement in data aggregation, since sensor nodes are typically deployed in unsecured locations and are not equipped with tamper-resistant hardware. An adversary is able to replay, modify, delay, drop, and deliver protocol messages out of order as well as inject own messages. However, most aggregation protocols assume that all intermediate nodes are trusted [3–14, 16–19, 21–23] except [1, 2, 15, 20]. A corrupted node can easily modify its own sensor reading. It is difficult to detect such a dishonest act in data aggregation,

[1] This study is partly supported by Grant-in-Aid for Scientific Research (A) (21240001), Grant-in-Aid for Young Scientists (B) (22700066), and IT Specialist Program of Ministry of Education, Culture, Sports, Science and Technology, Japan (MEXT).

since the modified sensor reading is indistinguishable from the legitimate reading. Such a dishonest act is called *direct data injection attack* [1, 15], where even one small modification might influence a total aggregated value. It is, thus, important to minimize the damage by direct data injection attacks. Such a security model is called *optimal security* [1, 20] (See Definition 1). We also employ the same security model. Optimal security means that the harmful influence on the final aggregation result is proportional to only the number of corrupted nodes which perform direct data injection attacks.

It is important to achieve constant congestion in large-scale wireless sensor networks. Sensors are usually resource-limited and power-constrained. They suffer from restricted computation, communication, and power resources. The energy savings of performing in-network aggregation are crucial for energy-constrained sensor networks. Since the nodes with the heaviest traffic are typically the nodes which are most essential to the connectivity of the network, their failure may cause the network to partition. Although several protocols [1, 2, 20] satisfy optimal security against multiple corrupted nodes, the congestion of these protocols is $\mathcal{O}(\log n)$ where n is the total number of sensor nodes.

In this paper, we propose an efficient and optimally secure sensor network aggregation protocol against multiple corrupted nodes by a *weak adversary*. Our protocol achieves one round-trip communication between each node and the base station to satisfy optimal security without the result-checking phase, by conducting aggregation along with the verification based on the idea of TESLA technique. Furthermore, we show that the congestion (maximum amount of per-node communication) in our protocol is constant. In other words, the amount of the per-node communication does not increase even if the number of nodes becomes huge. This means that our protocol suits large-scale wireless sensor networks.

The rest of this paper is organized as follows. In the next Section 2 we review a survey of other approaches to secure aggregation in sensor networks. Some requirements and preliminary items are provided in Section 3. We review the CPS protocol in Section 4. We propose our protocol in Section 5 and discuss the security and efficiency analysis of our protocol in Section 6. We finally conclude this paper in Section 7.

2 Related Work

There has been many works on preserving integrity in aggregation protocols. The simplest approach is a single-aggregator model [7–10] where each node sends its sensor reading directly to the aggregator (e.g., base station), and then the aggregator computes the final aggregation result. However, these protocols suffer from having a single node with high congestion. Also, several protocols do not assume corrupted nodes that are trying to disturb the aggregation result [11–14].

Recently several researchers have examined security issues in aggregation. Although some aggregation protocols [5, 6] are (optimally) secure against a single corrupted node without corruption of intermediate nodes, these protocols are not optimally secure against multiple corrupted nodes. Some aggregation protocols [1, 2, 20] are optimally secure against multiple corrupted nodes even if intermediate nodes are corrupted. These protocols addressed the issue of measuring and bounding corrupted node's contribution to the final aggregation result. In these protocols [1, 2, 20] related to our protocol, a network forms an aggregation tree, and then each node sends the aggregate up its parent node in the aggregation tree. The commitment is generated for the aggregate in a manner similar to a Merkle tree [24]. The schemes [16, 17] enhance the availability of the above schemes [1, 2], but do not discussed optimal security which is only discussed in [1, 20]. Wagner [15] performed a quantitative study measuring the effect of direct data injection attack on various aggregates.

Chan, Perrig and Song [1] defined optimal security for the first time. The CPS protocol uses two kinds of trees: aggregation tree and commitment tree. The commitment tree can be converted to a virtual binary tree for efficiency. As a result, the congestion for commitment verification is minimized.

Manulis and Schwenk [20] designs the data aggregation protocol in wireless sensor networks, called MS protocol, that satisfies optimal security. They provide a rigorous proof of optimal security against node corruption for the first time. The MS protocol aggregates all children data and sends it to parent node. It has two round-trip communications between each node and the base station including the result-checking phase, similar to the CPS protocol [1]. While the CPS protocol can convert an arbitrary tree to a binary commitment tree, the MS protocol does not consider such a conversion. As a result, the congestion of the MS protocol [20] is less efficient compared with the CPS protocol.

There have been several protocols introduced for preserving the confidentiality of aggregate results [18, 19, 10, 21–23]. This issue is orthogonal to our protocol and is not considered in this paper.

3 Preliminaries

3.1 Requirements

The following requirements need to be considered when designing secure in-network aggregation in wireless sensor networks.

- **Optimal security [1, 20]**. Optimal security is the concept to minimize the damaging impact of corrupted nodes on the overall aggregation result and assume the integrity of only data except for data modified by direct data injection attacks. The total aggregation result is modified only as long as the direct data injection attack is performed. It is usually difficult to find direct data injection attacks, and hence it is important not to expand the damage of direct data injection attacks.
- **Low congestion**. As a metric for communication overhead, we usually consider node congestion which is the worst case communication load on any single sensor node during the algorithm. Since the nodes with the heaviest traffic are typically the nodes which are most essential to the connectivity of

the network, their failure may cause the network to partition. Thus, lower communication traffic on the nodes with the heaviest traffic is desirable. Especially, node congestion should not depend on the total number of sensor nodes in large-scale wireless sensor networks.

- **Small number of communication rounds**. The communication between sensor nodes is not so reliable, owing to resource-limited and power-constrained. Thus, one round-trip communication for aggregation between each node and the base station is desirable, i.e., each node has only to send the aggregation messages to its parent node after receiving the query by the base station.
- **Low computational and storage costs**. A sensor node suffers from restricted computation and storage, hence the small computational and storage costs of a node are required. Especially, such costs should not depend on the total number of sensor nodes in large-scale wireless sensor networks. Of course, a node supports only the lightweight operations such as hash functions and symmetric-key encryption.

3.2 Network Assumptions

A sensor network might contain hundreds or thousands of tiny sensor nodes which suffer from restricted computation, communication, and power resources. Most architecture also employs more powerful base station, which is in one-to-many association with sensor nodes. We assume a general multi-hop network with a set $S = \{s_1, \ldots, s_n\}$ of n sensor nodes and a single trusted base station (BS). The sensor network is mostly static with a topology known to BS. This appears to be true of many modern sensor network applications such as a building management.

We also assume that aggregation is performed over an aggregation tree, which is the directed tree formed by the union of all the paths from the sensor nodes to BS (See Section 5). An aggregation transaction begins by broadcasting a query down the tree from BS to the leaves. Then, the sensor nodes measure their environment, and send their measurements back up the tree to BS. A large building with a control network that regulates inside temperatures by measuring the temperature in each room is one example of a hierarchical structure described in [15].

Each node can evaluate both the inputs and outputs of aggregation as mentioned in [20], defined as *Boolean predicates.* Restricting each sensor to read a value $v_i \in [v_{min}, v_{max}]$, the *inputs predicate* outputs *true* if and only if $v_{min} \leq v_i \leq v_{max}$. This means that a sensor node can evaluate the readings of other nodes. In the case of SUM aggregate, consequently, the *output predicate* outputs *true* if and only if $nv_{min} \leq a \leq nv_{max}$, where a is some intermediate aggregation result and n is the total number of sensor nodes which have already contributed into a. For instance, when each sensor node s_i senses temperature in the room, we may set the legitimate sensed value as $v_i \in [0, 50]$ (°C).

3.3 Security Infrastructure

For the purpose of authentication we consider that every sensor node s_i is in possession of some secret key denoted k_i shared between s_i and BS. Each sensor node s_i has a unique identifier I_i. We assume that the sensor nodes have the ability to perform computations of a collision-resistant cryptographic hash function H and secure message authentication code $\mathrm{MAC}_K(\cdot)$ where K is the cryptographic secret key.

3.4 Adversary Model

The primary concern of this paper are *stealthy attacks* as defined in [7]. In this type of attack, the adversary controls one or more nodes, and the goal of the adversary is to cause BS to accept a false aggregate by stealth. We refer to nodes that deviate from the protocol (including benign failures) as faulty nodes. An adversary tries to not only inject its own messages (i.e., direct data injection attacks) but also replay or modify the message sent by s_i or BS. Furthermore, we consider node corruption. We do not assume any tamper-resistance property. Upon corrupting s_i, the adversary obtains full control over s_i and reveals all information kept in s_i including its secret key k_i. We do not consider denial-of-service (DoS) attacks where the goal of the adversary is to prevent BS from getting any aggregation result at all. Such an attack will easily expose the adversary's presence.

We assume a "weak adversary model", i.e., an adversary cannot obtain the secret keys of adjoining two nodes (e.g., a node s_i and its child node). In other words, if the adversary can compromise a node s_i, then he does not obtain the secret key of s_i's children. This means that the adversary can manipulate the commitment of neither s_i's children nor s_i's parent when s_i is compromised. Here, we define the direct data injection attack [1, 15] as follows.
Direct data injection attack: The attack which modifies the data readings reported by the nodes under its direct control, under the constraint that the inputs predicate outputs true, is called "direct data injection attack" [1]. If a secure aggregation scheme has Boolean predicates, it can limit the adversary's capability to perform direct data injection.

Optimal security is the concept to minimize the damaging impact of corrupted nodes on the overall aggregation result and assume the integrity of only data except for data modified by direct data injection attacks. Optimal security means that the harmful influence on the final aggregation result is proportional to only the number of corrupted nodes which perform direct data injection attacks. We can thus define an optimal level of aggregation security as follows:

Definition 1 (Optimal security [1]) *An aggregation algorithm is optimally secure if, by tampering with the aggregation process, an adversary is unable to induce the base station to accept any aggregation result which is not already achievable by direct data injection attacks.*

4 The CPS Protocol

The CPS protocol [1], which was proposed by Chan, Perrig and Song in 2006, computes the sum aggregate by three phases: *query-dissemination*, *aggregation-commitment* and *result-checking*. The protocol [2] is the modification of the CPS protocol. These schemes assume that the sensor network is mostly static, with a topology known to the base station (BS). The CPS protocol has been already proved to satisfy "optimal security" in [1]. The CPS protocol uses two kinds of trees: aggregation tree and commitment tree. The commitment tree can be converted to a virtual binary tree for efficiency, i.e., all nodes are set to leaf nodes in such a virtual binary tree.

4.1 Protocol Description

Query-dissemination phase. To initiate a query in the aggregation tree, BS originates a query request message and distributes it to the aggregation tree. The query request message contains an attached nonce N to prevent replay of messages belonging to a prior query. Note that this request message can be sent without an authenticated broadcast, as described in [17].

Aggregation-commitment phase. Every node calculates a message based on its own sensor reading, and sends it to its parent node. This message consists of $\langle \mathsf{count}, \mathsf{value}, \mathsf{commitment} \rangle$, where count is the number of nodes in the subtree rooted at a node, value is the sum of all node values in the subtree, and commitment is the cryptographic commitment tree over the data values and the aggregation process in the subtree. Let v_i be a sensor reading of a node s_i. For a leaf node s_i, the message has the format $\langle 1, v_i, h_i \rangle$, where $h_i = H(N||v_i||ID_i)$ [2]. For an intermediate node, the message has the format $\langle c, v, h \rangle$ with $c = \sum c_j$, $v = \sum v_j$ and $h = H(N||c||v||\ell_1||\ldots||\ell_q)$, where its children have the following messages $\ell_1, \ell_2, \ldots, \ell_q$, where $\ell_j = \langle c_j, v_j, h_j \rangle$. Note that the intermediate node regards its own sensor reading as the reading from its child. In other words, the intermediate node sets a virtual leaf node as its child node. This means that all nodes are deployed as real leaf nodes and virtual leaf nodes in a binary tree. Nodes store the messages from their children, which will be used in the next result-checking phase. This result-checking phase ends with BS receiving the final message, including the final aggregate and the final commitment.

Result-checking phase. This phase has the following three steps: dissemination, check and acknowledgement.

- **Dissemination**. BS disseminates the final message to the network in an authenticated broadcast. Every node uses this message to verify that its own sensor reading was aggregated correctly. A node s_i is provided with not only the final message but also the messages of its *off-path* nodes from its parent (s_p). s_i's off-path nodes are the set of all the siblings of the nodes on

[2] We employ $h_i = H(N||v_i||ID_i)$ to prevent replay of messages from a leaf node, instead of $h_i = ID_i$ described in [1].

the path from s_i to BS. These are forwarded across the aggregation tree: s_p provides every child s_i with the messages of s_i's siblings in the commitment tree (an intermediate node has two children in the commitment tree), along with every off-path message received from s_p.
- **Check**. Using all off-path messages, s_i recomputes the messages of all its ancestors in the aggregation tree all the way to BS, and compares the result to the final message provided by BS.
- **Acknowledgement**. If the check succeeds, then s_i acknowledges by releasing an authentication code: $MAC_{k_i}(N||OK)$, where OK is a unique message identifier and k_i is the secret key shared between s_i and BS. Leaf nodes send their *ack*s while intermediate nodes wait for *ack*s from all their children, compute the XOR of those *ack*s with their own *ack*, and forward the resultant aggregated *ack* to their parent. Finally, BS has received the aggregated *ack*. If this aggregated *ack* is valid, then BS declares the aggregation successful.

4.2 Drawbacks

The CPS protocol has the following drawbacks:

- The communication overhead on each node is large. The CPS protocol requires two round-trip communications between each node and BS (one round-trip for query-dissemination phase and aggregation-commitment phase, and another round-trip for the result-checking phase) to do one aggregation procedure. Especially, the result-checking phase has the congestion of $\mathcal{O}(\log n)$.
- The computational cost at each sensor node is great. Not only BS but also each sensor node has to compute the final commitment in order to verify the integrity of its own sensor reading in the result-checking phase. Especially, a leaf node has the computational cost of $\mathcal{O}(\log n)$ in the result-checking phase.

4.3 Checking Mechanism

The result-checking phase enables the CPS protocol to satisfy "optimal security". All the honest nodes can check the validity of the own sensor reading in this phase after the final aggregation result is committed. This implies that the adversary cannot modify the sensor readings of honest nodes.

5 Our Protocol

5.1 Underlying Idea

Our goal is to do aggregation with one round-trip communication in wireless sensor networks. Our protocol achieves one round-trip communication by conducting aggregation along with the verification, based on the idea of basic tool of TESLA [25]. Also, a nonce is unnecessary in our protocol owing to one-time MAC-key to each session, and thus it protects our protocol against replay attacks. Note that the verification delay happens because of the TESLA-like verification done by the next session, although our protocol assumes a scene that regularly needs the aggregation.

Since our protocol uses *hash chain* technique in TESLA, the number of message authentication is restricted. However, such restriction is not so significant problem because the theft of sensor nodes increases with time and moreover the sensor node works on batteries. We assume that the operating time of the sensor node is innately limited.

5.2 The Format of Message

The message in our protocol consists of $\langle$count, value, identifier, commitment, confirmation$\rangle$, where count and value are the same as the CPS protocol. The identifier of node s_i is computed as $I_i = k_{i,0} = H^{\ell}(k_{i,\ell})$, where ℓ is the maximum number of sessions and $k_{i,\ell}$ is the secret key of the node s_i, shared between s_i and BS. Then, $k_{i,t} = H^{\ell-t}(k_{i,\ell})$ is the secret key of MAC at session t $(1 \leq t \leq \ell - 1)$ and is also treated as a kind of identifier of s_i at the next session $t+1$, which can be verified by $I_i \stackrel{?}{=} H^t(k_{i,t})$. The commitment is the cryptographic commitment and the confirmation is the confirmation result of MAC in the previous session. For a node s_i and its descendant nodes $\{s_j\}$, the message at session t has the following message format:

$$M_{i,t} = \langle c, v, k_{i,t-1}, \mu_{k_{i,t}}, (\bigoplus_j \lambda_{k_{j,t}}) \oplus \lambda_{k_{i,t}} \rangle, \tag{1}$$

$$\text{with} \quad \mu_{k_{i,t}} = MAC_{k_{i,t}}(c||v) \quad \text{and} \quad \lambda_{k_{j,t}} = MAC_{k_{j,t}}(p_{j,t-1}),$$

where c and $v = v_{i,t} + \sum_j v_{j,t}$ are the total number of nodes and the total value of sensor readings in the subtree rooted at this node s_i, respectively. $v_{j,t}$ is a sensor reading of s_j and $p_{j,t} \in \{OK, NG\}$ is the verification result of MAC of s_j at session t. Note that if a node s_i is a leaf node then $c = 1$, $v = v_{i,t}$ and the confirmation $= \phi$ (null). Also, if the children node $(s_{j_1}, \ldots, s_{j_q})$ are all nodes on leaves then we set $c = q + 1$ including the number of s_i. Even if $k_{i,t-1}$ is exposed from $M_{i,t}$, the secret key $k_{i,t}$ of MAC cannot be computed from $k_{j,t-1}$ owing to one-wayness of H. An intermediate node s_i sends its own message and forwards its children's messages to its parent. These messages have the format: $\langle M_{i,t},\ M_{j_1,t},\ \ldots,\ M_{j_q,t} \rangle$ with s_i's message $M_{i,t}$ and its children messages $M_{j_1,t}, \ldots, M_{j_q,t}$.

5.3 Protocol Description

Our protocol starts with query-dissemination phase, which is the same as the CPS protocol, then ends with aggregation-commitment phase. It does not need the result-checking phase. In aggregation-commitment phase, two steps of confirmation and aggregation are executed. If s_i is a leaf node then it executes neither confirmation process nor aggregation process. s_i has only to send the own message M_i to its parent. Here, we assume that an intermediate node s_i with a secret key $k_{i,\ell}$ has a set of its descendant nodes $\{s_j\}$ and its children and grandchildren nodes $\{s_m\} \subset \{s_j\}$. Let t be the number of a current session. We show only the aggregation-commitment phase since the query-dissemination is the same as the CPS protocol.

Example (Aggregation-commitment phase) :

- **Confirmation**. Let s_3 be a present intermediate sensor node at session t deployed in Fig.1. s_3 has preserved the information $\langle$count, value, identifier, commitment$\rangle$ of s_4, s_5, s_6 and s_7 of the previous session $t-1$. The node s_3 receives four messages $M_{4,t}$, $M_{5,t}$, $M_{6,t}$ and $M_{7,t}$ at session t. We consider the confirmation of s_6 by s_3 in this example, similar to s_4, s_5 and s_7. At first, s_3 verifies the identifier $k_{6,t-1}$ by $k_{6,t-2} \stackrel{?}{=} H(k_{6,t-1})$. If the verification of this identifier is valid then s_3 verifies the previous commitment $\mu_{k_{6,t-1}}$. If the verification of four commitments is valid then s_3 computes $\lambda_{k_{3,t}} = MAC_{k_{3,t}}(OK)$ and then computes $\bigoplus_{j=3}^{7} \lambda_{k_{j,t}}$ to include $M_{3,t}$. After computing the confirmation, s_3 discards $\langle$count, value, identifier, commitment$\rangle$ of s_4, s_5, s_6 and s_7 of the previous session $t-1$.
- **Aggregation**. s_3 uses these four messages to check three sensor readings $v_{4,t}$, $v_{5,t}$ and $v_{6,t}$. For checking the range of $v_{6,t}$, the node s_3 computes the difference between $(v_{6,t} + v_{7,t})$ in $M_{6,t}$ and $v_{7,t}$ in $M_{7,t}$. If $v_{min} \leq v_{4,t}, v_{5,t}, v_{6,t} \leq v_{max}$, then s_3 computes its own message:

$$M_{3,t}{=}\langle 5,\ \textstyle\sum_{j=3}^{7} v_{j,t},\ k_{3,t-1},\ MAC_{k_{3,t}}\big(5 || \sum_{j=3}^{7} v_{j,t}\big),\ \bigoplus_{j=3}^{7} MAC_{k_{j,t}}(p_{j,t-1})\rangle, \tag{2}$$

  where $k_{3,t-1} = H(k_{3,t})$. Then, s_3 sends its own message $M_{3,t}$ and forwards its children's messages $M_{4,t}$, $M_{5,t}$ and $M_{6,t}$ to its parent node s_1. When s_3's task is completed, s_3 preserves the information, $(1, v_4, k_{4,t-1}, \mu_{k_{4,t}})$, $(1, v_5, k_{5,t-1}, \mu_{k_{5,t}})$, $(2, v_6, k_{6,t-1}, \mu_{k_{6,t}})$ and $(1, v_7, k_{7,t-1}, \mu_{k_{7,t}})$ until the next session $t+1$. Note that s_3 need not forward $M_{7,t}$ to s_1 since $v_{7,t}$ is included in $M_{6,t}$ in Fig.1.

Protocol (Aggregation-commitment phase)

- **Confirmation**. The node s_i has preserved the information $\langle$count, value, identifier, commitment$\rangle$ of $\{s_m\}$ at the previous session $t-1$. When s_i receives messages from its children at session t, these messages contain both children's and grandchildren's messages. At first, s_i verifies the identifier $k_{m,t-1}$ by $k_{m,t-2} \stackrel{?}{=} H(k_{m,t-1})$ of $\{s_m\}$. If the verification of all identifiers of $\{s_m\}$ is valid then s_i verifies the previous commitments $\mu_{k_{m,t-1}}$ of $\{s_m\}$. If the verification of all commitments of $\{s_m\}$ is valid then s_i computes $\lambda_{k_{i,t}} = MAC_{k_{i,t}}(OK)$ and then computes $(\bigoplus_m \lambda_{k_{m,t}}) \oplus \lambda_{k_{i,t}}$ to include $M_{i,t}$. Otherwise, s_i computes the confirmation as $MAC_{k_{i,t}}(NG)$. After computing such confirmation, s_i discards $\langle$count, value, identifier, commitment$\rangle$ of the previous session $t-1$.

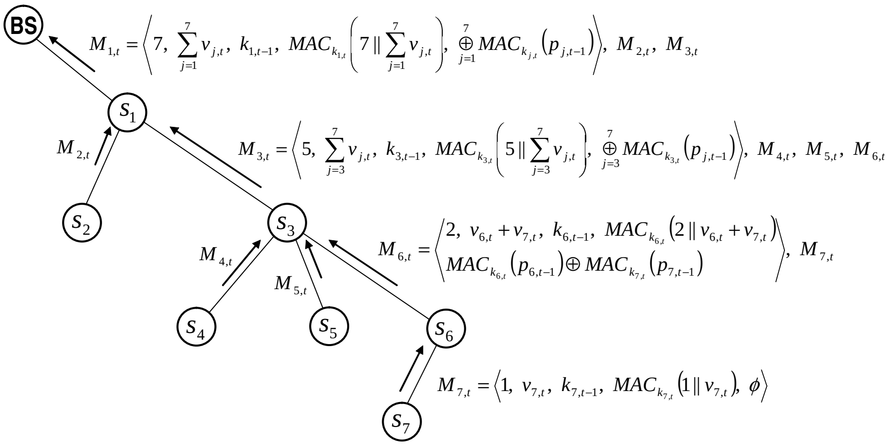

Fig. 1. Example of our protocol (session t).

– **Aggregation**. At first, s_i checks the sensor readings of children nodes using Boolean predicates (See Section 3.2). If s_i's child is a leaf node then s_i can directly check whether its child's sensor reading is in the range of $[v_{min}, v_{max}]$. Otherwise, s_i can check the range of its child's sensor reading by computing the difference between its child's aggregate and its grandchildren's aggregates. If sensor readings of s_i's children are out of range, then s_i rejects it. Otherwise, s_i computes its own message, which includes the aggregate at s_i (root of the subtree). Of course, s_i can obtain the values from its children's messages. Then, s_i sends its own message and forwards its children's messages to its parent. However, the node s_i need not forward its grandchildren's messages to its parent, because grandchildren's information is included in their children's messages. When s_i's task is completed, s_i preserves the information $\langle \mathsf{count}, \mathsf{value}, \mathsf{identifier}, \mathsf{commitment} \rangle$ of $\{s_m\}$ until the next session $t+1$. Finally, BS checks whether its children's sensor readings are in the range of $[v_{min}, v_{max}]$ in the same way as an intermediate node when BS has received the final messages. BS can compute the final commitments and the final confirmation using the secret key of each node. BS compares the computed confirmation with the received confirmation. If these results match, then BS accepts the aggregation result of the previous session $t-1$. Furthermore, if the final commitments are valid, then BS preserves the aggregation result until the next session. Otherwise, BS rejects the result.

A node receives the messages of its children and grandchildren at most. This means that the congestion of a node s_i is proportional to only the total number of $\{s_m\}$.

Remark. While the CPS protocol checks the validity of the own sensor readings in the result-checking phase, our protocol checks the validity of the children sensor readings in the aggregation-commitment phase. In our protocol, although the node s_i cannot verify the commitment of children and grandchildren at once, we achieves one round-trip communication by conducting aggregation along with the verification, based on the idea of TESLA.

6 Analysis

In this section, we discuss security and efficiency of our protocol. In the security analysis, we prove that our protocol is optimally secure. In the efficiency, we show that the congestion (maximum amount of per-node communication) in our protocol is constant. Also, we show that the computational cost of each node in our protocol is smaller than the CPS protocol.

6.1 Security

As mentioned in Section 3.4, an adversary should have only a limited influence on the result of the aggregation computation. We show that the SUM aggregate of our protocol satisfies "optimal security" described in Definition 1, similar to the CPS protocol. We show the following lemmas before showing theorems about "optimal security". If the values are in the range of $[v_{min}, v_{max}]$, then the values can be shifted to make a range of the form $[0, r]$.

Lemma 1 *Let v_a be a sensor reading of a node s_a. If s_a's parent accepts v_a then $0 \leq v_a \leq r$ is satisfied in our adversary model.*

Proof. Suppose s_p, v_p $\{s_b\}$ and $\sum v$ are s_a's parent node, s_p's sensor reading, a set of s_a's children and the sum of $\{s_b\}$'s aggregates, respectively. If s_a is honest and s_p accepts v_a, then $0 \leq v_a \leq r$ is naturally satisfied. We consider only the case that s_a is corrupted and s_p is honest. If s_a is a leaf node then s_p can easily check v_a. In this case, if s_p accepts v_a then $0 \leq v_a \leq r$. If s_a is an intermediate node then s_p can check v_a by computing the difference between $(v_a + \sum v)$ and $\sum v$, where $(v_a + \sum v)$ is s_a's aggregate. Hence if s_p accepts v_a then $0 \leq v_a \leq r$. Consequently, if s_p accepts v_a then $0 \leq v_a \leq r$. □

Lemma 2 *Suppose there are m nodes with committed (honest) sensor values $v_1, \ldots, v_m$ in an aggregation subtree T_m. Then the total of aggregation values by honest nodes at the root of T_m is $\sum_{i=1}^{m} v_i$.*

Proof. We show the result of three generations: a similar reasoning applies for arbitrary m nodes. Suppose s_p, v_p $\{s_h\}$ and $\sum v$ are s_a's parent node, s_p's sensor reading, a set of s_a's honest children and the sum of $\{s_h\}$'s aggregates, respectively. When the aggregate $\sum v$ is sent to s_p at session j, s_a cannot modify $\sum v$ because s_a does not know $\{s_h\}$'s secret keys. As a result, the legitimate aggregation $\sum v$ is included in s_p's aggregate even if s_a is computed. Therefore, the total of aggregation values by honest nodes at the root of T_m is $\sum_{i=1}^{m} v_i$. □

Theorem 1 *Let the final SUM aggregate received by BS be S. If BS accepts S then $S_L \leq S \leq (S_L + \mu r)$ where S_L is the sum of the data values of all the legitimate nodes, μ is the total number of malicious nodes, and r is the upper bound on the range of allowable values on each node.*

Proof. Let $s_1, \ldots, s_\mu$ be the nodes compromised by an adversary. The compromised node s_i $(1 \leq i \leq \mu)$ cannot manipulate the aggregates of s_i's children but it can manipulate its own sensor reading v_i. The sensor reading v_i satisfies $0 \leq v_i \leq r$ by Lemma 1. Hence it satisfies $0 \leq \mu(v_1 + \cdots + v_\mu) \leq \mu r$. Since S_L should be included in the total of aggregation values by Lemma 2, the SUM result S satisfies $S_L \leq S \leq (S_L + \mu r)$. □

Theorem 2 *Our protocol is optimally secure.*

Proof. Let the sum of the data values of all the legitimate nodes be S_L. Consider an adversary with μ malicious nodes which perform direct data injection attacks. An adversary causes the nodes under its control to report a sensor reading within the legal range $[0, r]$. If the adversary sets all μ nodes to have data value 0, the computed aggregate is S_L. If the adversary sets all μ nodes to have data value r, the computed aggregate is $S_L + \mu r$. Any aggregation value between these two extremes is achievable by both attacks. So, the bound of $S_L \leq S \leq (S_L + \mu r)$ by Theorem 1 is exactly on the range of possible results achievable by both attacks. Therefore, our protocol is optimally secure by Definition 1. □

6.2 Congestion Complexity

We now consider the congestion induced by the secure SUM aggregate. Congestion is a big problem for large-scale wireless sensor networks, so it is necessary to decrease congestion complexity. Our protocol aims to achieve the constant node congestion not to depend on the total number of nodes (n) owing to one round-trip communication without the result-checking phase. More specifically, we aim to stabilize the maximum amount of information that a single node sends and receives.

While the congestion in the CPS protocol is $\mathcal{O}(\log n)$ (strictly $\mathcal{O}(\log^2 n)$ in [1] and $\mathcal{O}(\log n)$ in [2]), the congestion of our protocol is constant, i.e., the congestion of a node s_a is proportional to only the total number of s_a's children and s_a's grandchildren. This means that our protocol is more scalable and efficient than the CPS protocol. Note that we do not mention the protocol [20] in our efficiency analysis since its congestion is less efficient compared with the CPS protocol.

Node congestion in our protocol eventually depends on the number of children nodes. Hence, the smaller the number of children nodes becomes, the lower the node congestion complexity gets in our protocol. Therefore, if the aggregation tree is composed not to grow the number of children then our protocol is more effective. Note that node congestion in our protocol does not depend on the height of aggregation tree.

Table 1. The communication complexity and computational cost of each node at each session.

		Communicationl complexity		Computational cost	
		Leaf node	**Intermediate node**	**Leaf node**	**Intermediate node**
CPS	AC	$\|H\|$	$\|H\|(\alpha+1)$	H	H
	RC	$\|M\|$	$2\|H\|\log n_{lower}+\|M\|$	$H\log n+M$	$H\log n_{upper}+M$
Ours	AC	$\|H\|+\|M\|$	$(\|H\|+2\|M\|)(2\alpha+\beta+1)$	$H+M$	$(H+M)(\alpha+\beta+1)+M$

6.3 Communication Complexity

One round-trip communication between each node and BS for aggregation is desirable. While the CPS protocol required two round-trip communications, our protocol indeed requires only one round-trip communication. Here, we explain the communication complexity of an intermediate node (A leaf node is also included in Table 1.). Let n_{lower} be the number of nodes which are deployed in lower part (only descendant nodes) of a current intermediate node ($n_{lower} \leq n$), and let $|H|$ and $|M|$ be the size of H and MAC, respectively. In the aggregation-commitment phase (AC), while the size of message in the CPS protocol is almost $|H|(\alpha+1)$ which is mainly the size of commitments that a node sends and receives, the size of message in our protocol is almost $(|H|+2|M|)(2\alpha+\beta+1)$ which is mainly the size of messages a node sends and receives, described in Table 1, where α and β are the number of children and grandchildren nodes, respectively. However, the size of message additionally requires $2|H|\log n_{lower}+|M|$ in the result-checking phase (RC) of the CPS protocol. Therefore, the communication complexity of our protocol is smaller than that of the CPS protocol in large-scale wireless sensor networks.

6.4 Computational and Storage Costs

The computational and storage costs should not depend on the total number of sensor nodes n because of its restricted computation and storage. Here, we explain the computational cost of an intermediate node (A leaf node is also included in Table 1.). We compare our protocol with the CPS protocol by the per-node computational cost described in Table 1. Let n_{upper} be the number of nodes which are deployed in upper level of a current intermediate node ($n_{upper} \leq n$), and let H and M be the computation of H and MAC, respectively. Both the CPS protocol and our protocol employ only the lightweight computation like hash function. However, the computational cost of the CPS protocol depends on the number of nodes n_{upper}, i.e., $H\log n_{upper}+M$ (a leaf node has the worst case of computational cost of $H\log n+M$), while that of our protocol does not depends on the total number of nodes, i.e., it is the constant value $(H+M)(\alpha+\beta+1)+M$ in the aggregation-commitment phase (AC). Thus, our protocol is more effective than the CPS protocol for the per-node computational cost in large-scale wireless sensor networks.

The size of extra storage in our protocol is $(|H|+|M|)(\alpha+\beta)$ compared with the CPS protocol, because a node has to preserve the messages of its children and grandchildren of the previous session. Note that a leaf node need not have such storage. However, it is not so significant problem in large-scale wireless sensor networks since the size of storage is constant, i.e., it does not depend on n.

6.5 Other Operations

The SUM aggregate is easily used to compute the total number of votes in the network, where all the nodes have value either 1 or 0. Also, the average can be easily computed by dividing the SUM aggregate by the total number of nodes. Furthermore, the proposed protocol can use the verification of Φ-quantile aggregate, as described in [1].

7 Conclusion

We proposed an efficient and optimally secure sensor network aggregation protocol for general networks and multiple corrupted nodes. Our protocol satisfies "optimal security" which guarantees that the harmful influence on the final aggregation result is proportional to only the number of corrupted nodes which perform direct data injection attacks. As a result, the influence on the total aggregate can be optimally controlled within a certain range. Furthermore, since our protocol is one round-trip communication without the result-checking phase, both the node congestion complexity and the computational cost of each node are constant in our protocol. Therefore, our protocol suits large-scale wireless sensor networks.

References

1. H. Chan, A. Perrig and D. Song, "Secure hierarchical in-network aggregation in sensor networks," Proc. 13th ACM Conference on Computer and Communications Security – CCS'06, pp.278–287, ACM, 2006.
2. K.B. Frikken and J.A. Dougherty, "An efficient integrity-preserving scheme for hierarchical sensor aggregation," Proc. 1st ACM Conference on Wireless Network Security – WiSec'08, pp.68–76, ACM, 2008.
3. S. Madden, M. Franklin, J. Hellerstein and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," Proc. ACM SIGOPS Operating Systems, pp.131–146, ACM, 2002.
4. Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," Proc. ACM SIGMOD Record, vol.31, pp.9–18, ACM, 2002.
5. L. Hu and D. Evans, "Secure aggregation for wireless networks," Proc. 2003 Symposium on Applications and the Internet Workshops – SAINT'03, pp.27–31, IEEE, 2003.

6. P. Jadia and A. Mathuria, "Efficient secure aggregation in sensor networks," Proc. High Performance Computing – HiPC'04, LNCS 3296, pp.40–49, Springer-Verlag, 2004.
7. B. Przydatek, D. Song, and A. Perrig, "SIA: Secure information aggregation in sensor networks," Proc. 1st International Conference on Embedded Networked Sensor Systems – SenSys'03, pp.255–265, ACM, 2003.
8. W. Du, J. Deng, Y.S. Han and P. Varshney, "A Witness-Based Approach for Data Fusion Assurance in Wireless Sensor Networks," Proc. IEEE Global Telecommunications Conference – GLOBECOM'03, pp.1435–1439, IEEE, 2003.
9. A. Mahimkar and T.S. Rappaport, "SecureDAV: a secure data aggregation and verification protocol for sensor networks," Proc. IEEE Global Telecommunications Conference – GLOBECOM'04, pp.2175–2179, IEEE, 2004.
10. Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks," Proc. 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing – MobiHoc'06, pp.356–367, ACM, 2006.
11. I. Gupta, R. Van Renesse and K.P. Birman, "Scalable fault-tolerant aggregation in large process groups," Proc. International Conference on Dependable Systems and Networks – DSN'01, pp.433–442, IEEE, 2001.
12. S. Nath, P.B. Gibbons, S. Seshan and Z.R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," Proc. 2nd International Conference on Embedded Networked Sensor systems – SenSys'04, pp.250–262, ACM, 2004.
13. J.Y. Chen, G. Pandurangan and D. Xu, "Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis," Proc. 4th International Symposium on Information Processing in Sensor Networks – IPSN'05, pp.348–355, IEEE, 2005.
14. A. Manjhi, S. Nath and P.B. Gibbons, "Tributaries and deltas: efficient and robust aggregation in sensor network streams," Proc. 2005 ACM SIGMOD International Conference on Management of Data, pp.287–298, ACM, 2005.
15. D. Wagner, "Resilient aggregation in sensor networks," Proc. 2nd ACM Workshop on Security of Ad-hoc and Sensor Networks – SASN'04, pp.78–87, ACM, 2004.
16. P. Haghani, P. Papadimitratos, M. Poturalski, K. Aberer and J.P. Hubaux, "Efficient and robust secure aggregation for sensor networks," Proc. 3rd Workshop on Secure Network Protocols – NPSec'07, pp.1–6, IEEE, 2007.
17. H. Chan and A. Perrig, "Efficient security primitives derived from a secure aggregation algorithm," Proc. 15th ACM Conference on Computer and Communications Security – CCS'08, pp.521–534, ACM, 2008.
18. W. Zhang, C. Wang and T. Feng, "GP2S: Generic Privacy-Preservation Solutions for Approximate Aggregation of Sensor Data," Proc. IEEE International Conference on Pervasive Computing and Communications – PerCom'08, pp.179–184, IEEE, 2008.
19. C. Castelluccia, E. Mykletun and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," Proc. 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services – MOBIQUITOUS'05, pp.109–117, IEEE, 2005.
20. M. Manulis and J Schwenk, "Security model and framework for information aggregation in sensor networks," ACM Transactions on Sensor Networks (TOSN), vol.5, no.2, ACM, 2009.
21. D. Westhoff, J. Girao and M. Acharya, "Concealed data aggregation for reverse multicast traffic in sensor networks: encryption, key distribution, and routing adap-

tation," IEEE Transactions on Mobile Computing, vol.5, no.10, pp.1417–1431, IEEE, 2006.
22. W. He, X. Liu, H. Nguyen, K. Nahrstedt, T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks, " Proc. 26th IEEE International Conference on Computer Communications – INFOCOM'07, pp.2045–2053, IEEE, 2007.
23. S.Q. Ren, D.S. Kim and J.S. Park, "A secure data aggregation scheme for wireless sensor networks," Proc. Frontiers of High Performance Computing and Networking – ISPA'07, LNCS 4743, pp.32-40, Springer-Verlag, 2007.
24. R.C. Merkle, "Protocols for public key cryptosystems," Proc. IEEE Symposium on Security and Privacy – IEEE S&P, pp.122–134, IEEE, 1980.
25. A. Perrig, R. Canetti, J.D. Tygar and D. Song, "Efficient Authentication and Signing of Multicast Streams over Lossy Channels," Proc. IEEE Symposium on Security and Privacy – IEEE S&P, pp.56–73, IEEE, 2000.