An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks

Sencun Zhu[1] Sanjeev Setia[1*] Sushil Jajodia[1,2]
[1]Center for Secure Information Systems
George Mason University
Fairfax, VA 22030
[2]The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
{*szhu1,setia,jajodia*}*@gmu.edu*

Peng Ning
Computer Science Department
North Carolina State University
Raleigh, NC 27695
pning@ncsu.edu

Abstract

Sensor networks are often deployed in unattended environments, thus leaving these networks vulnerable to false data injection attacks *in which an adversary injects false data into the network with the goal of deceiving the base station or depleting the resources of the relaying nodes. Standard authentication mechanisms cannot prevent this attack if the adversary has compromised one or a small number of sensor nodes. In this paper, we present an interleaved hop-by-hop authentication scheme that guarantees that the base station will detect any injected false data packets when no more than a certain number t nodes are compromised. Further, our scheme provides an upper bound B for the number of hops that a false data packet could be forwarded before it is detected and dropped, given that there are up to t colluding compromised nodes. We show that in the worst case B is $O(t^2)$. Through performance analysis, we show that our scheme is efficient with respect to the security it provides, and it also allows a tradeoff between security and performance.*

1. Introduction

Consider a military application of sensor networks for reconnaissance of the opposing forces, as shown in Fig. 1. Suppose we want to monitor the activities of the opposing forces, e.g., tank movements, ship arrivals or departures, and other relevant events. To achieve this goal, we can deploy a cluster of sensor nodes around each area of interest. We can then deploy a base station in a secure location to control the sensors and collect data reported by the sensors. To facilitate data collection in such a network, sensor nodes on a path from an area of interest to the base station can relay the data to the base station.

The unattended nature of the deployed sensor network lends itself to several attacks by the adversary, including physical destruction of sensor nodes, security attacks on the routing and data link protocols, and resource consumption attacks launched to deplete the limited energy resources of the sensor nodes.

Unattended sensor node deployment also makes another attack easier: an adversary may compromise several sensor nodes, and then use the compromised nodes to inject false data into the network. This attack falls in the category of *insider attacks*. Standard authentication mechanisms are not sufficient to prevent such insider attacks, since the adversary knows all the keying material possessed by the compromised nodes. We note that this attack can be launched against many sensor network applications, though we have only given a military scenario.

In this paper, we present a scheme for addressing this form of attack, which we call a *false data injection attack*. Our scheme enables the base station to verify the authenticity of a report that it has received as long as the number of compromised sensor nodes does not exceed a certain threshold. Further, our scheme attempts to filter out false data packets injected into the network by compromised nodes before they reach the base station, thus saving the energy for relaying them.

In a recent work, Przydatek, Song, and Perrig proposed SIA [16], a secure information aggregation scheme for sensor networks that addresses a similar problem to ours. SIA addresses the issue of false data injection using statistical techniques and interactive proofs, ensuring that the aggre-

* also with Computer Science Dept, George Mason University

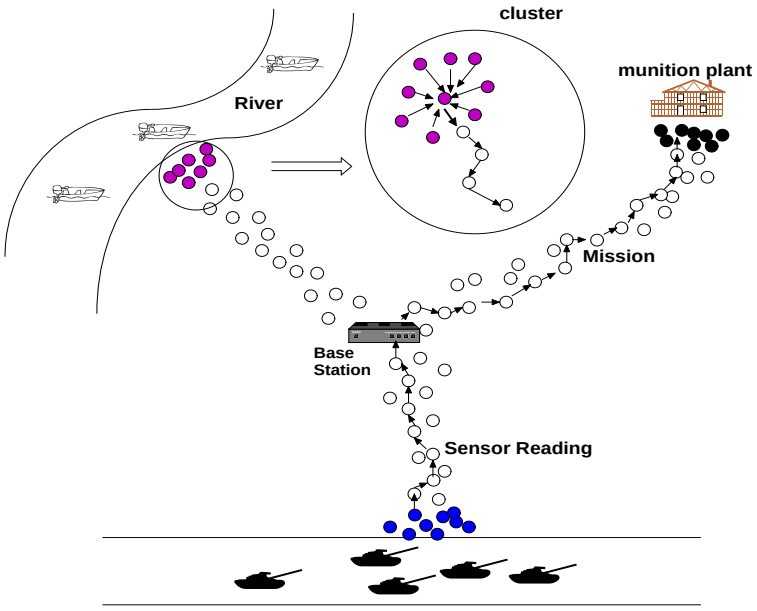

Figure 1. An example sensor network. Suppose we want to monitor three areas of interest, the road, the river, and the munition plant, by deploying a cluster of sensor nodes (filled circles) in each area. The base station sends commands or queries to the sensor nodes, and receives reports from them. All the communications are relayed by some forwarding nodes (blank circles).

gated result reported by the aggregation node (the base station) is a good *approximation* to the true value, even if a small number of sensor nodes and the aggregation node may have been compromised. In contrast, the focus of our work is on *detecting and filtering out* false data packets, either at or en-route to the base station. Our scheme is particularly useful for large-scale sensor networks where a sensor report needs to be relayed over several hops before it reaches the base station, and for applications where the information contained in the sensor reports is not amenable to the statistical techniques used by SIA (e.g., non-numeric data). We note that our scheme and SIA address complementary problems, and the techniques of both schemes can be combined to make the network more robust to false data injection attacks.

To defend against false data injection attacks, we present an authentication scheme in which at least $t + 1$ sensor nodes have to agree upon a report before it is sent to the base station. Further, all the nodes that are involved in relaying the report to the base station authenticate the report in *an interleaved, hop-by-hop* fashion. Here t is a security threshold based on the security requirements of the application under consideration and the network node density. Our scheme guarantees that if no more than t nodes are compromised, the base station will detect any false data packets injected by the compromised sensors. In addition, for a given t, our scheme provides an upper bound B for the number of hops that a false data packet can be forwarded before it is detected and dropped. If every noncompromised node on the path between a cluster head and the base station knows the ids of the nodes that are $t + 1$ hops away from it on the path, then $B = t$; otherwise, without this knowledge, $B = (t - 1)(t - 2)$. Through performance analysis, we show that our scheme is efficient with respect to the security it provides.

The remainder of this paper is organized as follows. Section 2 introduces our scheme in detail. In Section 3 and Section 4 we study the security and the performance of our scheme. We introduce the related work in Section 5, and conclude our work in Section 6.

2. An Interleaved Hop-by-hop Authentication Scheme

2.1. Assumptions

We describe the assumptions regarding sensor networks before we present our scheme in detail.

Network and Node Assumptions Sensor nodes can be deployed via aerial scattering or by physical installation. We assume that in an area of interest, sensor nodes are organized into clusters. Each cluster includes at least $t + 1$

nodes, where t is a design parameter. In a cluster, one node is elected to be the cluster head, and each cluster has a unique cluster id. The issues of electing a node as the cluster head and how to generate a unique cluster id are out of the scope of this paper. A cluster head collects sensor readings or votes from $t+1$ cluster nodes (including itself), and then reports the result to the base station. Note that the role of cluster head may rotate among the cluster nodes, according to an appropriate criteria such as remaining energy.

We assume network links are bidirectional; that is, if node u can hear node v, node v can also hear node u. Sensor nodes are similar to the current generation of sensor nodes (e.g. the Berkeley MICA motes [10]) in their computational and communication capabilities and power resources. We assume that every node has space to store several hundred bytes of keying materials.

Security Assumptions We assume that every node shares a master secret key with the base station. We also assume that each node knows (at least a subset of) its one-hop neighbors, and has established a pairwise key with each of them. We argue that this is a reasonable assumption. For example, we can use the pairwise key establishment scheme in LEAP [19] to achieve this goal. Under this assumption, the impact of a node compromise is *localized* in the immediate neighborhood of the compromised node. We further assume that a node can establish a pairwise key with another node that is multiple hops away, if needed. For example, if the network size is small (for example, fewer than 200 nodes), we can employ either the Blom scheme [3] or the Blundo scheme [4] directly. For a larger network, we may use the extensions [6, 15] to these schemes to tolerate a possibly larger number of node compromises. In all these schemes, two nodes only need to know each other's id to establish a pairwise key, and the computational overhead is shown to be affordable for current generation sensor nodes [6, 15]. For simplicity, we refer to these schemes as *id-based* schemes. Since we mention the Blundo scheme frequently as an example of an id-based scheme during the description of our scheme, we provide a brief introduction to this scheme in Appendix A.

We further assume that the base station has a mechanism to authenticate broadcast messages (e.g., based on μTESLA [17]), and every node can verify the broadcast messages. Because the role of cluster head may rotate among cluster nodes, we assume that all nodes are equally trusted. We assume that if a node is compromised, all the information it holds will also be compromised. However, we assume that the base station will not be compromised.

2.2. Threat Model and Design Goal

Since wireless communication is broadcast-based, we assume that an adversary can eavesdrop on all traffic, inject packets, and replay older packets. We assume that an adversary can take full control of compromised nodes. Thus, an adversary may command compromised nodes to drop or alter messages going through them, aiming at preventing the base station from receiving authentic sensor readings.

In this paper, we focus on *false data injection attacks*, in which an attacker's goal is to cause false alarms or to deplete the already-constrained resources of forwarding nodes by injecting false data. We assume that the compromised nodes can collude in their attacks. Our goal is to design an authentication scheme that can defend against false data injection attacks launched by up to t compromised nodes, where t is a system parameter. This scheme should have the following properties when there are no more than t compromised nodes. First, the base station should be able to detect any false data packet injected by a compromised node. Second, the number of hops before an injected data packet is detected and discarded should be as small as possible. Third, the scheme should be efficient in computation and communication with respect to the security it provides. Finally, the scheme should be robust to node failures.

2.3. Notation and Definition

Notation The following notations appear in the rest of this discussion.

- u, v (in lower case) are principals such as communicating nodes.
- K_u is the key of node u shared with the base station.
- K_{uv} is the pairwise key shared between nodes u and v.
- G is a family of pseudo-random functions [8].
- K_u^a is node u's authentication key, derived as $K_u^a = G_{K_u}(0)$.
- $MAC(k, s)$ is the message authentication code (MAC) of message s generated with a symmetric key k.

We denote the base station as BS and the head of a cluster of sensor nodes as CH. Let n be the number of hops between BS and CH, and u_i ($1 \leq i \leq n$) be an intermediate node on the path from CH to BS, where i increases from CH to BS. Let v_i ($1 \leq i \leq t$) denote one of the t cluster nodes other than CH in a cluster.

Definition 1 *For two nodes u_i and u_j on the path from CH to BS, if $|i-j| = t+1$, we say u_i and u_j are* associated, *and u_i is an associated node of u_j. More specifically, if $i - j = t+1$, u_i is the* upper associated node *of node u_j, and u_j is the* lower associated node *of node u_i.*

From this definition, we know that a node that is less than $t+1$ hops away from BS does not have an upper associated

node. Also note that an intermediate node may have multiple lower associated nodes if it has multiple child nodes leading to multiple clusters. We further extend this definition by including the following two special cases.

- A node u_i ($1 \leq i \leq t$) that is less than $t+1$ hops away from CH has one of the cluster nodes v_i ($1 \leq i \leq t$) as a lower associated node.
- The cluster head CH is associated with u_{t+1}.

Fig. 2 shows a node cluster and a path from the cluster head to the base station, where $t = 3$. Node u_3 has an upper associated node u_7 and a lower association node v_3. Node u_5 has a lower associated node u_1 but no upper associated node.

2.4. Scheme Overview

Our scheme involves the following five phases:

1. In the *node initialization and deployment* phase, the key server loads every node with a unique id, as well as necessary keying materials that allow the node to establish pairwise keys with other nodes. After deployment, a node first establishes a one-hop pairwise key with each of its neighbors.
2. In the *association discovery* phase, a node discovers the ids of its associated nodes. This process may be initiated by the base station periodically, or by a node that detects the failure of a neighbor node.
3. In the *report endorsement* phase, $t + 1$ nodes generate a report collaboratively when they detect the occurrence of an event of interest. More specifically, every participating node computes two MACs over the event, one using its key shared with the BS, and the other using its pairwise key shared with its upper associated node. Then it sends the MACs to its cluster head. The cluster head collects MACs from all the participating nodes, wraps them into a report, and then forwards the report towards BS.
4. In the *en-route filtering* phase, every forwarding node verifies the MAC computed by its lower association node, and then removes that MAC from the received report. If the verification succeeds, it then computes and attaches a new MAC based on its pairwise key shared with its upper associated node. Finally, it forwards the report to the next node towards the BS.
5. In the *base station verification* phase, the BS verifies the report after receiving it. If the BS detects that $t + 1$ nodes have endorsed the report correctly, it accepts the report; otherwise, it simply discards the report.

2.5. The Basic Scheme

In this section, we illustrate the basic idea in our scheme. We will discuss it in more detail in Sections 2.6 and 2.7.

2.5.1. Node Initialization and Deployment The key server loads every node with a unique integer id, ranging from 0 to the maximal number of nodes in the network. Therefore, for example, a node id is of size two bytes if the number of nodes in the network is between 256 and 65536. The key server also loads every node u with necessary keying materials. Specifically, it pre-loads node u with an individual key K_u shared with the base station. From K_u, node u derives its authentication key K_u^a. If the one-hop pairwise key establishment scheme in LEAP [19] is employed, node u is loaded with an initial network key. If the Blundo scheme [4] is used for establishing multi-hop pairwise keys, the key server randomly generates a symmetric bivariate polynomial of degree k, and loads node u with the $k + 1$ coefficients of polynomial $f(u, y)$. After node u is deployed, it discovers all its one-hop neighbors and then establishes a pairwise key with each of its neighbors.

2.5.2. Association Discovery The *association discovery* phase is necessary for a node to discover the ids of its association nodes. We first describe a two-way association discovery scheme for the initial path setup, which consists of two steps – *base station hello* and *cluster acknowledgment*. We then describe an incremental association discovery scheme in Section 2.6, which is executed if the upper and/or lower associated nodes of a node change because of changes in the path from a cluster to the base station. We also discuss some variants of the scheme in Section 2.7.

Base Station Hello This step enables a node to discover its upper association node. The base station initiates this process by broadcasting a HELLO message, which is recursively forwarded to all nodes so that every node discovers the ids of the $t + 1$ closest nodes that are on its path to the base station. On receiving a HELLO message from the base station, a node attaches its own id to the HELLO message before re-broadcasting it. Our scheme restricts the maximum number of node ids that are included in a message to $t + 1$. To achieve this, each node replaces the id of the node that is $t + 1$ hops closer to the base station with its own id. Thus, the communication overhead introduced by a HELLO message is bounded by $t + 1$ ids, despite the number of hops the HELLO message travels. On receiving the HELLO message, a cluster head assigns each of the $t + 1$ ids in the message to one of its cluster nodes (including itself). In addition, if a cluster head is also an en-route node for another cluster, it will rebroadcast the HELLO message.

Fig. 3 shows an example where $t = 3$. BS is the base station and CH is the cluster head of a cluster C_i that con-

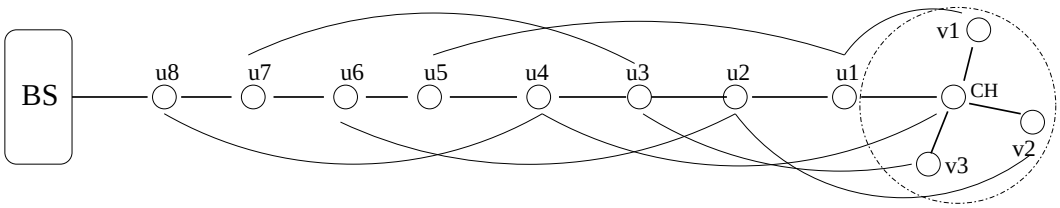

Figure 2. An example showing the definition of *association* where $t = 3$. BS is the base station and CH is a cluster head. Two nodes connected with an arc are associated, the one closer to the base station is the *upper* associated node and the other is the *lower* associated node.

sists of nodes v_1, v_2, v_3, and CH. BS broadcasts a HELLO message M, which includes its id BS and a sequence number S_n. Here S_n is used to prevent replay attacks as well as message loops. M is authenticated by an authentication scheme such as μTELSA.

After receiving M, node u_6 records the id(s) in M, attaches its own id to M, and then rebroadcasts M. Nodes u_5 and u_4 do the same. When M arrives at node u_3, M already contains $t + 1 = 4$ node ids. Node u_3 records S_n and the ids in M, removes the first id (here BS) in the id list, adds its own id to the end of the id list, and then rebroadcasts M. Nodes u_2 and u_1 also do the same. When node CH, the cluster head, receives M, it assigns the ids of the preceding nodes to its cluster nodes. For example, it assigns u_3 to v_3, u_2 to v_2 and u_1 to v_1, respectively. Thus, u_1, u_2, and u_3 are associated with v_1, v_2, and v_3, respectively, and CH is associated with u_4. At the end of this step, every node that is more than $t + 1$ hops away from BS has an upper association node.

Cluster Acknowledgment After the base station hello step, the cluster head sends back an acknowledgment ACK back to the base station. The ACK includes the cluster id, and the ids of the $t + 1$ lower association nodes. When a node receives the ACK, it will check if all the node ids in the ACK are distinct. If not, it will drop the ACK (we will explain the reason in security analysis in Section 3). During the forwarding of the ACK, the node ids are replaced in the opposite direction in the base station hello step (i.e., a node removes the last id in the id list and adds its own id in the beginning), thus allowing a receiving node to learn the id of its lower association node. In the case that a node has multiple child nodes leading to multiple clusters, it has multiple lower associations. Therefore, it maintains a table that includes multiple path information, where each path is uniquely identified by the corresponding cluster id. Moreover, because the cluster acknowledgment message is critical for a node to maintain correct association knowledge, we can employ a hop-by-hop acknowledgment mechanism to avoid packet losses due to unreliable link layer transmissions.

Consider Fig. 3. The cluster header CH first computes a MAC over S_n and the cluster id C_i, using its authentication key K^a_{CH}. CH then generates an acknowledgment, which includes its id CH, the above MAC, and an ordered list of ids of the $t + 1$ cluster nodes that have discovered their upper associated nodes in the base station hello phase. CH sends the acknowledgment to u_1, the node that previously forwarded the HELLO message to CH. The id list in the acknowledgment message is $\{CH, v_3, v_2, v_1\}$. As a result, u_1 discovers that its lower association is v_1, the last one in the list. Node u_1 then removes v_1 from the list and inserts its own id at the beginning of the list. The id list it sends to u_2 is then $\{u_1, CH, v_3, v_2\}$. In this way, every node on the path discovers its lower association node, while the size of the acknowledgment message remains bounded.

During this phase, every node stores the id list it receives. Moreover, the acknowledgment is authenticated in a hop-by-hop fashion; that is, every node authenticates the acknowledgment to its up-stream node using their pairwise key as the MAC key. When the base station receives the acknowledgment, it verifies the acknowledgment and records the id of the cluster. We will discuss the security of this procedure in more detail in Section 3.

2.5.3. Report Endorsement Sensor nodes generate a report when triggered by a special event, e.g., an increase in the temperature being monitored by the nodes, or in response to a query from the base station. Our scheme requires that at least $t + 1$ nodes agree on the report for it to be considered a valid report. For example, at least $t + 1$ neighboring nodes should agree that the local temperature is higher than $150F$ for a valid report to be sent to the base station. Thus, if $t > 0$, an adversary cannot cause a false fire alarm by compromising just one sensor node.

When a node v agrees on an event E (E typically contains an event type and a timestamp), it computes a MAC for E, using its authentication key K^a_v as the MAC key. In addition, node v computes another MAC for E, using the pairwise key shared with its upper association node u as the MAC key. Note that both u and v can compute their pairwise key K_{uv} based on an id-based pairwise key establish-

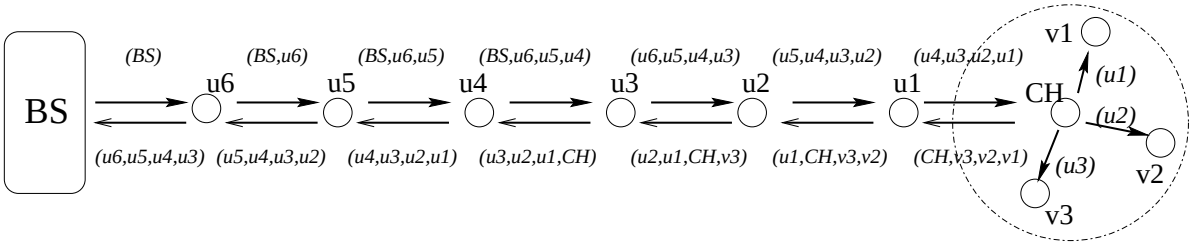

Figure 3. An example illustrating the base station hello step where $t = 3$. BS is the base station, u_i is an en-route node. CH is the cluster head and v_1, v_2, v_3 are cluster nodes. (M) is the content of the beaconing message. Note that u_i may be an en-route node for multiple paths and CH may also be an en-route node for another cluster, although we only show one path in this figure.

ment scheme. We refer to these two types of MACs as *individual MAC* and *pairwise MAC*, respectively. Node v then sends an endorsement message to the cluster head that includes these two MACs. The cluster head collects endorsements from $t + 1$ cluster nodes (including itself). It then compresses the $t + 1$ individual MACs by XORing them to reduce the size of a report. However, the pairwise MACs are not compressed for transmission, because otherwise a node relaying the message will not be able to extract the pairwise MAC of interest to it. The cluster head finally generates a report, which contains the event E, a list of ids of the endorsing nodes, the compressed MAC and $t + 1$ pairwise MACs. We will discuss the use of a short pairwise MAC to reduce the message overhead in Section 4.

Consider the cluster node v_1 in Fig. 4. v_1 computes two MACs over the event E; one MAC key is its authentication key $K^a_{v_1}$ and the other is the pairwise key $K_{v_1u_1}$ shared with its upper associated node u_1. v_1 sends its endorsement that contains these two MACs to the current cluster head CH. The endorsement is authenticated with the pairwise key shared between v_1 and CH.

CH collects endorsements from the other two nodes v_2 and v_3 as well. It then verifies the authenticity of each endorsement based on its pairwise key shared with the corresponding cluster node. If all the endorsements are authenticated, CH computes a compressed MAC over E, denoted as $XMAC(E)$.

$$XMAC(E) = MAC(K^a_{v_1}, E) \oplus MAC(K^a_{v_2}, E) \oplus MAC(K^a_{v_3}, E) \oplus MAC(K^a_{CH}, E).$$

The report R that node CH finally generates and forwards towards BS is as follows.

$$R: \quad E, C_i, \{v_1, v_2, v_3, CH\}, XMAC(E), \{MAC(K_{CHu_4}, E), MAC(K_{v_3u_3}, E), MAC(K_{v_2u_2}, E), MAC(K_{v_1u_1}, E)\}.$$

The report includes the ids of the endorsing nodes v_1, v_2, v_3, CH, which allows the base station to verify the compressed MAC later. These ids may be removed in future reports to save bandwidth overhead unless the nodes in the endorsing set have changed, since the base station can identify the endorsing nodes from the cluster id C_i. The order of the pairwise MACs in R corresponds to that in the cluster acknowledgment message so that a node receiving R knows which pairwise MAC is from its lower association node. Moreover, R is authenticated with the pairwise key shared between CH and the next node on the path.

2.5.4. En-route Filtering When a node u receives R from its downstream node, it first verifies the authenticity of R based on its pairwise key shared with that node. Then it checks the number of different pairwise MACs in R. If node u is $s (s < t + 1)$ hops away from BS, it should see s pairwise MACs; otherwise, it should see $t + 1$ pairwise MACs. It then verifies the last MAC in the pairwise MAC list, based on the pairwise key shared with its lower association node. In the case that it has not computed the pairwise key earlier, it computes the pairwise key and then stores it. Note that node u will drop the report if any of the above checks fails. If node u is more than $t + 1$ hops away from BS, it proceeds to computes a new pairwise MAC over event E using the pairwise key shared with its own upper association node. It then removes the last MAC from the MAC list and inserts the new MAC into the beginning of the MAC list. Finally it forwards the report to its upstream node.

Consider node u_1 in Fig. 4. When node u_1 receives the report R from node CH, it checks if there are 4 pairwise MACs. If true, it computes its pairwise key shared with node v_1, $K_{u_1v_1}$, if it has not computed $K_{u_1v_1}$ before. Node u_1 then verifies the last MAC in R, $MAC(K_{v_1u_1}, E)$. If the verification succeeds, node u_1 computes a new MAC over E, using the pairwise key it shares with node u_5. The output is $MAC(K_{u_1u_5}, E)$. Finally, node u_1 inserts the $MAC(K_{u_1u_5}, E)$ into the beginning of the MAC list, and removes the last MAC on the list, $MAC(K_{v_1u_1}, E)$. The

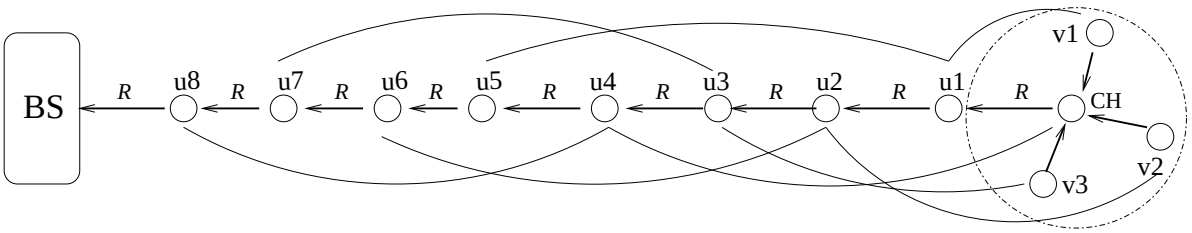

Figure 4. An example of report endorsement and en-route filtering where $t = 3$.

report R that node u_1 forwards to node u_2 is as follows (R is also authenticated with $K_{u_1u_2}$).

$$R: \quad E, C_i, \{v_1, v_2, v_3, CH\}, XMAC(E), \{MAC(K_{u_1u_5}, E), MAC(K_{CHu_4}, E)\}, MAC(K_{v_3u_3}, E), MAC(K_{v_2u_2}, E).$$

All the other forwarding nodes perform the same steps. However, the nodes within $t + 1$ hops of BS do not insert a new pairwise MAC. It is very easy to see that every node on the path from the cluster head to the base station can verify one pairwise MAC in the report independently in addition to the MAC computed by its direct downstream node. Thus the report is authenticated in an interleaved hop-by-hop fashion.

2.5.5. Base Station Verification The base station BS only needs to verify the compressed MAC. Basically, it computes $t + 1$ MACs over E using the authentication keys of the nodes in the id list, then XORs the MACs to see if it matches the one in the report. The BS can easily compute the authentication key of a node based on its id. If the report is authenticated and BS knows the location of every cluster node, it can locate these reporting nodes and then react to the event. On the other hand, if the verification fails, BS will discard the report.

2.6. Association Maintenance

The correctness of our scheme relies on correct association knowledge. A node needs to know the id of its lower association node; otherwise, it will not know which pairwise key to use to verify a pairwise MAC. In addition, it needs to know the id of its upper association node so that it can add a valid pairwise MAC into a report; otherwise, its upper association node will drop the report. If the path between the base station and a cluster head is static, then only an initial association discovery process is necessary. However, if the path between the base station and a cluster head changes due to the failure of an intermediate node or other reasons, our scheme has to adapt to the change accordingly to maintain correct associations. We discuss below association maintenance in two scenarios, namely *base station initiated repair* and *local repair*.

2.6.1. Base Station Initiated Repair In this scenario, once a path is formed, the reports from a cluster head to the base station always follow the same path, unless the path is changed due to the base station. For example, in the TinyOS beaconing protocol [10], the base station broadcasts a beaconing message periodically forming a breadth-first tree rooted at the base station. Specifically, every node records its parent node as the node from which it first receives the beaconing message during the current epoch, and then rebroadcasts the beaconing message. Thus, the path between a cluster head and the base station is changed when an intermediate node chooses different parent nodes in two consecutive time epochs.

To adapt to path changes, our scheme can execute the base station hello step for each epoch by piggybacking node ids in every beaconing message. The cluster acknowledgment process can also be omitted by letting a lower association node enclose its id with its pairwise MAC when it forwards a report. This strategy works well for networks where the topology changes frequently at the additional bandwidth expense of $t + 1$ ids per beaconing message. For less dynamic networks, this overhead should be reduced. Especially, if a path does not change during different epochs, it is not necessary for a node to attach its id to the beaconing message.

We adopt a reactive approach for association maintenance in relatively static networks. Recall that in the base station hello step, every node records s ids that are the ids of the nodes that are on its path to the base station. Here $s = t + 1$ if a node is more than t hops away from the base station; otherwise s is the actual number of hops from the base station. A node can infer that its own s upstream nodes are unchanged if it receives a beaconing message from the same parent node and the beaconing message is in its original format (i.e., no node ids are added), and if a node forwards the original beaconing message only if its own $s - 1$ upstream nodes are unchanged. We can see that if a path is unchanged during different epochs, our scheme will not incur any additional bandwidth overhead. However, when a node selects a parent node that is different from the one

in the previous epoch, it sends a request to the new parent node to get the ids of $s-1$ upstream nodes, and then attaches these $s-1$ node ids and its own id to the beaconing message it is forwarding.

2.6.2. Local Repair In the base station initiated repair scheme, if the underlying routing protocol has a large beaconing period, the failure of an intermediate node on a path may cause many reports to be dropped. Therefore, it is necessary for the nodes detecting the failure of a neighbor to locally repair a path that avoids the failed node. This will result in inconsistent node association relationship in our scheme. Thus, we need a scheme to locally repair the association relationship adaptively.

Our approach for local repair is based on the right-hand rule in the greedy parameter stateless routing (GPSR) protocol [11]. Here we assume every node knows the locations or relative locations of its neighbors (e.g., because of GPS). Fig. 5 illustrates the approach. When node u_4 detects its parent node u_5 has failed (the issue of node failure detection is out of our scope), it sends a REPAIR message to w_1, which is the first node counterclockwise about u_4 from edge (u_4, u_5). The REPAIR message includes the ids of $t+1$ upstream nodes of u_4 except u_5; that is, it includes the id list $\{u_6, u_7, u_8\}$. w_1 and w_2 forward the REPAIR message based on the same rule. When w_3 receives the message, it finds that its next node is u_6 based on this rule and u_6 is in the list $\{u_6, u_7, u_8\}$, which means that the failed node has been bypassed. w_3 then requests an id list from u_6, which includes the ids of t upstream nodes of node u_6 and u_6. Node w_3 then forwards the list to its downstream nodes in the same way as in the base station hello step. Thus, every node whose associations have been changed reestablishes its association relationship proactively.

Note that although a local repair process is necessary to maintain path connectivity, it is also very important to limit the frequency with which the process is invoked. Otherwise, a compromised node may invoke this process very frequently to consume the energy of the involved nodes. To thwart this attack, for instance, we can limit the number of invocations to be at most one within one beaconing epoch.

2.7. Interaction with Routing Protocols

The advantage of the two-way association discovery protocol described above is its independence from the underlying protocols, making it applicable for various sensor network applications. On the other hand, we note that the association discovery process usually overlaps with the route discovery process in a routing protocol. Therefore, in practice we can combine the association discovery protocol with the underlying routing protocol if it is beneficial. As described earlier, we can integrate the base station hello process with the TinyOS beaconing protocol [10] by piggybacking the ids of the upper association nodes in a beaconing message. As another example, if we want to adapt our scheme to the GPSR [11] protocol, in addition to piggybacking node ids, the base station should unicast (instead of broadcast) its HELLO messages to the next node towards the cluster head, based on the location of the cluster head.

3. Security Analysis

We discuss the security of our scheme with respect to our two design goals, i.e., the ability of the base station to detect a false report and the ability of the en-route nodes to filter false reports.

3.1. Base Station Detection

Our authentication scheme requires that each of $t+1$ cluster nodes compute an individual MAC based on its authentication key that is only shared with the base station. Thus, it guarantees that an adversary has to compromise at least $t+1$ nodes to be able to forge a report to deceive the base station. Note that our scheme compresses $t+1$ individual MACs into one MAC based on the bitwise XOR operation (instead of attaching $t+1$ individual MACs) to reduce message overhead. This compression scheme is secure because it is a special case of the XOR-MAC scheme [1] which is proven to be secure.

3.2. En-route Filtering

In this section, we discuss the en-route filtering capability of our scheme for two attack models, namely, *outsider attacks* launched by an adversary that has not compromised any nodes, and *insider attacks* launched by an adversary that has compromised up to t nodes.

3.2.1. Outsider Attacks In our scheme, every message is authenticated in a hop-by-hop fashion during its transmission. Thus, an unauthorized node cannot inject false data without it being detected. Moreover, because an event also contains a timestamp, an attack in which an outsider replays an old report will be detected.

3.2.2. Insider Attacks We consider several insider attacks by up to t compromised nodes. We first discuss the security of our scheme under the assumption that every node knows the authentic ids of its upper association and its lower association. This corresponds to the situations in which every node is loaded with correct association knowledge before it is deployed, or every node learns the ids of its association nodes through the association discovery process before any nodes are compromised. A compromised node can provide an authenticated pairwise MAC over any data to deceive its upper association node. Thus,

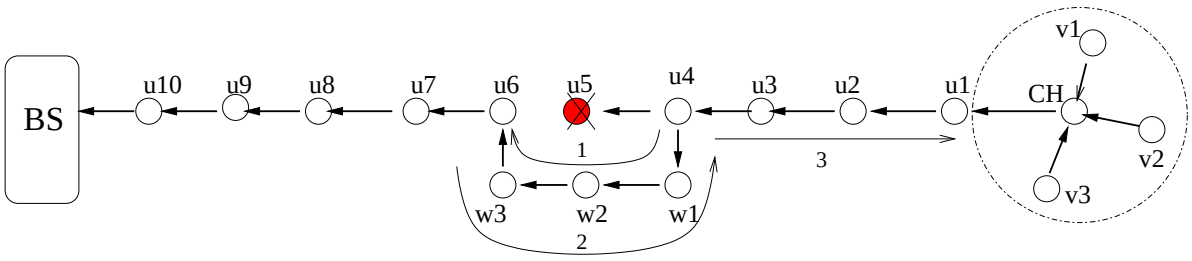

Figure 5. An example showing the local repair process when node u_5 fails and node u_4 establishes a new path towards BS. The new path includes nodes w_1, w_2 and w_3.

if totally t nodes are compromised, they can provide t authenticated pairwise MACs over a false report, which will bypass the verifications of t noncompromised upper association nodes. However, in our scheme every report must provide $t + 1$ pairwise MACs; therefore, one noncompromised node will filter out the false report because the pairwise MAC from its lower association node will be invalid. Thus, our scheme guarantees that a false report will be dropped after it is forwarded by at most t noncompromised nodes.

The above analysis indicates that the security of our scheme relies on the assumption of authenticated association knowledge. Thus, we need to analyze the security of the association discovery process that provides association knowledge to nodes. More specifically, the security of the *cluster acknowledgment* process is critical because it provides the lower association knowledge that is used as the basis for en-route filtering to the nodes. The cluster acknowledgment process is subject to attack if it is executed at any time after t nodes have been compromised.

Before we show two types of attacks on the cluster acknowledgment process, we first clarify the attack model. Recall that in the cluster acknowledgment phase, when a node u receives an acknowledgment message ACK from its downstream neighbor (authenticated with their pairwise key), it verifies the ACK and then checks if all the ids in the id list in the ACK are distinct. If the check is successful, it will set the id of its lower association node to the last id in the list. Then it removes the last id and adds its own id to the beginning of the list. The goal of an attack on this process is to *lower associate more than t noncompromised nodes to t compromised nodes*, under the constraint that $t + 1$ distinct ids must appear in the list when the ACK is forwarded. This attack is possible mainly because in a multi-hop pairwise key establishment process, two nodes do not know the actual number of hops between them. In other words, when a node u establishes a pairwise key with another node v, it trusts v only because v can compute the same secret key – it does not know where v is.

Cluster Insider Attacks In this attack, all the t compromised nodes are from the cluster (possibly including the cluster head); that is, no nodes on the path to the base station are compromised. Because the ACK from the cluster head towards the base station must contain $t + 1$ distinct node ids, it must include the id of a noncompromised or nonexistant node. Therefore, one of the $t + 1$ relaying nodes closest to the cluster head (e.g., node u_1, u_2, u_3 and u_4 in Fig. 4) will be lower associated to a noncompromised or nonexistant node. This node will drop a false report. Thus, we reach the same conclusion that a false report will be dropped after it is forwarded by at most t noncompromised nodes.

En-route Insider Attacks In this attack, t compromised nodes that lie on the path to the base station collude to attack the cluster acknowledgment process. The worst case scenario occurs when the cluster head CH and $t - 1$ forwarding nodes are compromised, and these t compromised nodes are equally separated by t noncompromised nodes. In other words, t compromised nodes isolate t^2 noncompromised nodes into t blocks of t nodes each. Let the ids of the nodes from CH to BS be

$$CH, \{u_{1,1}, u_{1,2}, ..., u_{1,t}\}, X_1, \{u_{2,1}, u_{2,2}, ..., u_{2,t}\}, \\ X_2, ..., X_{t-1}, \{u_{t,1}, u_{t,2}, ..., u_{t,t}\}, ..., BS.$$

Here the ids of the compromised nodes are $CH, X_1, X_2, ..., X_{t-1}$, and two consecutive compromised nodes are segregated by t noncompromised nodes $u_{i,j} (1 \leq i, j \leq t)$. The cluster head CH first forges a list of $t + 1$ ids, among which the first one is a randomly picked legitimate id y and the other t ids are the ids of the t compromised nodes. For example, the id list could be $\{y, CH, X_1, X_2, ..., X_{t-1}\}$. According to the cluster acknowledgment process, every noncompromised node between $u_{1,1}$ and $u_{1,t}$ sets its lower association node as the last id in the list, removes the last id and then inserts its own id to the beginning of the list. Thus, each of nodes $u_{1,1}, u_{1,2}, ..., u_{1,t}$ is lower associated to a compromised node. The id list node $u_{1,t}$ outputs is $\{u_{1,t}, ..., u_{1,2}, u_{1,1}, y\}$; therefore, the next en-route

node will be lower associated to y. If the next node has not been compromised yet, it will drop any false data immediately because it will not see a valid pairwise MAC provided by y. To resume the forwarding of the false report, the adversary has to compromise the next node, here X_1. X_1 does not forward this id list to its upstream neighbor $u_{2,1}$; instead, it forges the same or a similar id list as the one generated by CH. Thus, similarly each of $u_{2,1}, u_{2,2}, ..., u_{2,t}$ is lower associated with one compromised node. Repeatedly, we can easily see that totally t^2 nodes (t in each block) are associated with the same set of t compromised nodes. Thus, in this case, a false report will be dropped after it is forwarded by at most t^2 noncompromised nodes. We note, however, that this upper bound corresponds to the worst case; an adversary has to compromise t nodes selectively to achieve this upper bound.

Enhancements to the Basic Scheme The above conclusion that in the worst case t compromised nodes can collude to deceive t^2 noncompromised nodes is drawn based on the assumption that every node only checks if an id list includes $t+1$ distinct ids. We can further reduce this upper bound by adding more constraints that are easy to implement in practice.

First, we can exploit the fact that in our scheme each node knows its authenticated neighbor set. When an enroute node receives an id list in an acknowledgment message from a downstream node, it additionally checks if the downstream node is the first one in the list because in our scheme a node adds its own id to the beginning of an id list.

Consider the first block in the above worst case. Under this constraint the cluster head CH has to place CH at the beginning of the list and y as the second id in the list when it sends a false list to $u_{1,1}$. As a result, the adversary has to compromise the node that is t hops away from CH to regenerate a false list because y becomes the last one in the id list after the list is forwarded for $t-1$ hops. In other words, there are at most $t-1$ noncompromised nodes between them (i.e., no $u_{1,t}$). Thus, the upper bound t^2 is reduced to $t(t-1)$.

Furthermore, instead of simply adding its own id to the list, a node can add an id pair that includes its id and the id of its lower association. Consider the first block again. If the list which CH sends to $u_{1,1}$ includes X_1, then one of $u_{1,1}, u_{1,2}, ..., u_{1,t-1}$, say $u_{1,i} (1 \leq i \leq t-1)$, will have X_1 as its lower association node. Since a node also adds the id of its lower association node to the list, node $u_{1,t-1}$ will see the pair $\{u_{1,i}, X_1\}$ in the list. Because X_1 is its neighbor, $u_{1,t-1}$ knows that X_1 cannot be a lower association node of one of its downstream nodes. Therefore, to avoid being detected by $u_{1,t-1}$, the adversary will not include X_1 in the list in this block. Similarly, the adversary will not include the id of one of the compromised nodes in the list in other blocks. Thus, we achieve the upper bound $t(t-2)$.

Second, our scheme can add a node feedback mechanism to facilitate compromise detection. For example, after a node receives a certain number of false data packets, it will send an ALERT report to the base station. An ALERT report contains information such as the id of the node from which a node received false data. The base station can then take necessary actions such as traceback to identify the compromised node. Traceback is possible in our scheme because every packet is authenticated during its transmission and a compromised node is localized (see Section 3.3 for details). A compromised node may drop any ALERT report going through it, but it cannot force the nodes between itself and the base station to do the same. Thus, to avoid being detected for false data injection, the compromised node that is closest to the base station will need to drop the false data injected by a colluding compromised node.

Consider the above worst case scenario again. Node X_{t-1} will need to drop any false data injected by its coalition. Thus, the adversary can mount false data injection attacks to consume the energy of the relaying nodes between CH and X_{t-1} without being known by the base station. To this end, if the adversary wants to achieve this goal, it can at most deceive $(t-1)(t-2)$ noncompromised nodes to relay false data. For example, if $t = 4$, we have the upper bound 6; if $t = 5$, the upper bound is 12.

Third, if all sensor nodes possess GPS devices, our scheme can further reduce the upper bound greatly, at the expense of larger performance overhead. The idea is to embed the location of a node into its id such that a compromised node cannot lie about its location. The scheme makes the security assumption that a sensor node will not be compromised before it obtains its coordinate through GPS after its deployment. Given this assumption, we can adapt, for instance, the Blundo scheme [4] in the following way. First, the key server pre-loads every node u with a bivariate polynomial $f(x, y)$ of degree-k for both x and y (not $f(u, y)$, as in the original scheme). After its deployment, node u generates a new id u', which combines its id u and its coordinate (e.g., by simply concatenating them or hashing the concatenation into a shorter id). Then node u evaluates $f(u', y)$ itself, and then *erases* $f(x, y)$. Thus, when it is compromised later, it will not reveal any information about the polynomials which other nodes possess. When two nodes want to establish a pairwise key, they have to provide to each other the authentic information of their locations. Based on the coordinates and as well as the default transmission range, they will know the minimum number of hops between them. Furthermore, if the base station broadcasts its own coordinate, every node will know roughly the direction of the path it is located on. As a result, the adversary cannot use the a compromised node that is located between a node u and the base station or is more than t hops away from node u as a lower

association of node u. To this end, a compromised node normally can only deceive one noncompromised node; therefore, the upper bound is t or slightly larger than t.

Note that although this variation is very effective at preventing false data injection attacks, we do not consider t as the upper bound for our scheme mainly due to performance considerations. First, a node u needs to store $k^2 + 1$ coefficients for $f(x, y)$. Second, it needs to compute k^2+k modular multiplications. Third, the exchange of coordinates may consume a nontrivial portion of the bandwidth. Therefore, to trade off security for performance, we only consider this in the least resource constrained case.

3.3. Other Security Issues

A compromised yet undetected node can always drop or alter every packet going through it. There is no way to prevent it from doing so. The only solution is to detect the compromised node and then bypass it. Compromise detection in a sensor network is a very difficult issue, because a sensor network is usually deployed in an unattended environment. Due to the difficulty of compromise detection, the security bottom line of a security protocol for sensor networks is that the impact of a node compromise must be localized so as to provide the basis for later compromise detection. If a compromised node can only mount such attacks on its own behalf and the attacks can only occur around its initial deployment location, the node will take a great risk of being detected. Our scheme does meet the above security bottom line. Recall that our scheme starts with a node initialization and deployment process. After this phase, every node knows the authentic set of its direct neighbors and establishes a pairwise key with each of them, and it will *only* accept packets authenticated by one of the nodes from its neighbor set. This implies that a compromised node can only mount an attack *locally* and *on its own behalf*. Note that, on the other hand, since the primary design goal of our scheme is to prevent false data injection attacks, the above attacks actually do not conflict with our design goal because these attacks also lead to early packet dropping.

4. Performance Evaluation

In this section, we analyze the computational and communication overheads of our basic scheme.

4.1. Computational Cost

The computational overhead of our scheme arises mostly due to two operations – establishing pairwise keys and report authentication.

Establishing Pairwise Keys In our scheme, two association nodes need to establish a multi-hop pairwise key on the fly, based on one of the id-based schemes [3, 4, 6, 14, 15]. All these schemes have similar computational overhead. For example, in the Blundo scheme, a node needs to compute k modular multiplications and k modular additions for a polynomial of degree k. Let $k = 100$, and the size of a secret key be 64 bits and the size of a node id be 16-bits (assuming there are no more than $64K$ sensor nodes in a network). The cost of computing a pairwise key is about $1/10000$ of that of creating a RSA signature, which is of the same order as that of an AES encryption. Moreover, in our scheme normally a cluster node computes one pairwise key and an en-route node computes two. In the case of node failure or a path change, a node has to compute a pairwise key shared with a new node; however, this situation does not happen very frequently.

Report Authentication In our scheme, a cluster node computes three MACs for one report. One uses its individual key as the MAC key, the second uses a pairwise key shared with its association node as the MAC key, and the third uses the pairwise key shared with its cluster head as the MAC key. An en-route node normally computes four MACs — it verifies one pairwise MAC (over E) from its lower association and generates one pairwise MAC (over E) for its upper association if it is more than t hops away from BS; it verifies one MAC (over R) from its downstream neighbor and generates one MAC (over R) for its upstream neighbor. Note that although in our scheme a forwarding node computes two more MACs (over E) than that in a hop-by-hop authentication scheme [19], the security it achieves is much stronger. Since the energy for computing a MAC is about the same as that for transmitting one byte [18], by filtering false data as early as possible, our scheme reduces the overall energy expenditure of a node even though it entails additional computational costs.

4.2. Communication Cost

The communication overhead of our scheme arises from two sources. First, every authentic report contains one compressed MAC and $t + 1$ pairwise MACs. In practice, we can choose a larger size for an individual MAC, while selecting a smaller size for a pairwise MAC. The size of the compressed MAC must be large enough because the authenticity of an reported event is security critical. Since the size of a pairwise MAC only impacts the capability of en-route filtering, we can make it smaller as a tradeoff between performance and security. For example, if we use 4 bytes for pairwise MACs, and $t = 3$, the size of a pairwise MAC will be 1 byte. Compared to a standard hop-by-hop authentication scheme [19], our scheme introduces the additional

message overhead of 4 bytes in this example, but it also provides much stronger security.

Second, during association discovery and maintenance, a node adds its own id to a beaconing message when its path changes. Thus, this component of the communication overhead depends on path dynamics.

Finally, we mention that the choice of t should be based on both security and node density. A large t makes it more difficult for the adversary to launch a false data injection attack, but it also requires more nodes to form a cluster.

5. Related Work

Perrig et al [17] presented μTESLA for base station broadcast authentication, based on one-way key chains [13] and delayed key disclosure. Zhu et al [19] presented a scheme that is also based on one-way key chains for local (one-hop) broadcast authentication with the goal of enabling authenticated passive participation in sensor networks. Although this scheme is robust against outsider attacks, it is vulnerable to insider attacks in which an adversary only needs to compromise a single node to inject false data. In contrast, our interleaved hop-by-hop authentication scheme is robust to insider attacks involving a certain number of compromised nodes.

Hu and Evans [9] propose a secure hop-by-hop data aggregation scheme that works if one node is compromised (i.e., $t = 1$). Ye et al [18] propose a statistical en-route detection scheme called SEF, which allows both the base station and en-route nodes to detect false data with a certain probability. With an overhead of 14 bytes per report, SEF is able to drop $80-90\%$ of the injected false reports by a compromised node (i.e., $t = 1$) within 10 forwarding hops. In our scheme, when $t = 1$, a false data packet will be dropped immediately. Moreover, the packet overhead of our scheme is also smaller.

Przydatek et al [16] present SIA, a secure information aggregation scheme for sensor networks. As discussed in the introduction, both SIA and our work aim to defend against false data injection. The main difference between these two schemes is in the focus. SIA focuses on the accuracy of query results reported from the base station, whereas our scheme focuses on the authenticity of the reports from sensor nodes and provides a means to filter out any injected false data as early as possible. We believe the combination of our scheme with SIA would make a sensor network more robust to false data injection attacks.

Other work on sensor network security include studies on key management [5, 6, 7, 15, 19]. The polynomial-based pairwise key establishment scheme [4] has been recently extended [15] to enable a sensor network to sustain more node compromises under the same memory constraints. The idea is to use probabilistic polynomial pre-deployment (in a manner similar to probabilistic key pre-deployment [7]) so that every node is loaded with a random subset of polynomials from a large pool of polynomials.

6. Conclusion and Future Work

In this paper, we presented a simple but effective authentication scheme to prevent false data injection attacks in sensor networks. The scheme guarantees that the base station can detect a false report when no more than t nodes are compromised, where t is a security threshold. In addition, our scheme guarantees that t colluding compromised sensors can deceive at most B noncompromised nodes to forward false data they inject, where B is $O(t^2)$ in the worst case. Our performance analysis shows this scheme is efficient with respect to the security it provides and allows a tradeoff between security and performance.

Recently, we have improved the scheme presented in this paper such that a false data packet injected into the network will be detected within one hop, i.e., $B = 0$. This improvement is achieved at the expense of additional computational overhead per node, although the communication overhead of both schemes is identical. We refer the reader to [20] for a description of the improved scheme and an analysis of its security and performance in comparison to the scheme presented in this paper.

As future work, several directions are worth investigating. In particular, we plan to study the use of interleaved hop-by-hop authentication for preventing or mitigating attacks against sensor network routing and data collection protocols, such as those pointed out in [12]. Another topic that we plan to address is how our scheme can be adapted for sensor networks with mobile data sinks.

Acknowledgements

We thank Fan Ye for valuable discussions during the early stages of this work, and thank Jing Deng, Leijun Huang, and Sankardas Roy for their helpful comments. We also thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions. in Proc. of Crypto'95.

[2] S. Basagni, K. Herrin, E. Rosti, D. Bruschi. Secure Pebblenets. In Proc. of MobiHoc 2001.

[3] R. Blom. An Optimal Class of Symmetric Key Generation Systems. Advances in Cryptology, EUROCRYPT'84, LNCS 209, 335338, 1984.

[4] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic

conferences. In Advances in Cryptology CRYPTO 92, LNCS 740, pages 471486, 1993.

[5] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In Proc. of the IEEE Security and Privacy Symposim 2003, May 2003.

[6] W. Du, J. Deng, Y. Han, and P. Varshney. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In Proc. of 10th ACM Conference on Computer and Communications Security (CCS), Washington DC, October 27-31, 2003.

[7] L. Eschenauer and V. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In Proc. of 9th ACM Conference on Computer and Communications Security (CCS), Washington DC, 2002.

[8] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. Journal of the ACM, Vol. 33, No. 4, 1986, pp 210-217.

[9] L. Hu and D. Evans. Secure aggregation for wireless networks. In Workshop on Security and Assurance in Ad hoc Networks. Jan. 2003.

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

[11] B. Karp and H. Kung. GPSR: A Geographic Hash Table for Data-Centric Storage. In Proc. of ACM International Workshop on Wireless Sensor Networks and Applications, Sept. 2000.

[12] C. Karlof and D. Wagner. Secure Routing in Sensor Networks: Attacks and Countermeasures. In Proc. of First IEEE Workshop on Sensor Network Protocols and Applications, May 2003.

[13] L. Lamport. Password authentication with insecure communication communication. Communications of the ACM, 24(11):770-772, Nov., 1981.

[14] D. Liu and P. Ning. Location-Based Pairwise Key Establishments for Static Sensor Networks. In 2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03), October 2003.

[15] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October, 2003.

[16] B. Przydatek, D. Song, and A. Perrig. SIA: Secure Information Aggregation in Sensor Networks. In Proc. of ACM SenSys 2003.

[17] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. SPINS: Security Protocols for Sensor Networks. In Proc. of Seventh Annual ACM International Conference on Mobile Computing and Networks(Mobicom 2001), Rome Italy, July 2001.

[18] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical En-route Detection and Filtering of Injected False Data in Sensor Networks. To appear in Proc. of IEEE INFOCOM 2004.

[19] S. Zhu, S. Setia and S. Jajodia. LEAP: Effi cient Security Mechanisms for Large-Scale Distributed Sensor Networks. In Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October, 2003.

[20] 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. Technical Report, Dept. of Information and Software Engineering, George Mason University, March 2004.

Appendix A: The Blundo Scheme

The Blundo scheme was originally proposed by Blundo et al. [4] to allow any group of m parties to compute a common key while being secure against collusion between some of them. Here we use a special case of this scheme for establishing pairwise keys between two sensor nodes in the context of sensor networks.

The scheme works as follows. The key server first randomly generates a symmetric bivariate k-degree polynomial $f(x,y) = \sum_{i,j=0}^{k} a_{ij} x^i y^j$ over a finite field F_q, where q is a prime number that is large enough to accommodate a cryptographic key. A polynomial $f(x,y)$ is said to be symmetric if $f(x,y) = f(y,x)$. The key server computes $f(i,y)$ for node i, and then loads node i with all the $k+1$ coefficients (as a function of y). When two nodes i and j want to establish a pairwise key, they compute $f(i,j)$ (or $f(j,i)$, which is the same) by evaluating $f(i,y)$ at point j and $f(j,y)$ at point i, respectively. $f(i,j)$ serves as their pairwise key.

The above scheme has been proved to be unconditionally secure and k-collusion resistant [4]; that is, an adversary knows nothing about the pairwise key between any two non-compromised nodes if the number of sensor nodes it has compromised is no more than k. However, if the adversary compromises more than k nodes, it will know all the pairwise keys in the network. Therefore, it is important to choose a large enough degree k for the polynomial for the application under consideration. For the current generation sensor nodes, k can be around 200.