Interleaved Hop-by-Hop Authentication Against False Data Injection Attacks in Sensor Networks

SENCUN ZHU
The Pennsylvania State University
and
SANJEEV SETIA
George Mason University
and
SUSHIL JAJODIA
George Mason University
and
PENG NING
North Carolina State University

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. We present three interleaved hop-by-hop authentication schemes that guarantee that the base station can detect injected false data *immediately* when no more than t nodes are compromised, where t is a system design parameter. Moreover, these schemes enable an intermediate forwarding node to detect and discard false data packets as early as possible. Our performance analysis shows that our scheme is efficient with respect to the security it provides, and it also allows a tradeoff between security and performance. A prototype implementation of our scheme indicates that our scheme is practical and can be deployed on the current generation of sensor nodes.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; K.6.5 [**Management of Computing and Information Systems**]: Communication Networks—*Security and Protection*

General Terms: Security,Algorithm,Design

Additional Key Words and Phrases: Authentication, Filtering False Data, Interleaved Hop-by-Hop, Sensor Networks

A preliminary version of this paper appeared in Proceedings of IEEE Symposium on Security and Privacy, 2004.
Authors addresses: Sencun Zhu, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802. Sanjeev Setia and Sushil Jajodia, Center for Secure Information Systems, George Mason University, Fairfax, VA 22030. Peng Ning, Computer Science Department, North Carolina State University, Raleigh, NC 27695.
Emails: szhu@cse.psu.edu, {setia, jajodia}@gmu.edu, pning@ncsu.edu.

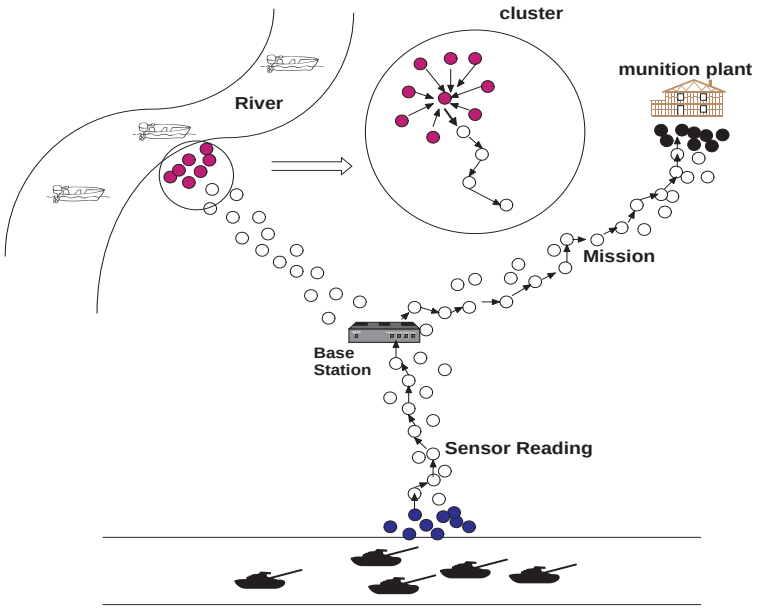

Fig. 1. An example sensor network. Suppose we want to monitor three areas of interest, the road, the river, and the munition plant, by deploying a cluster of sensor nodes (filled circles) in each area. The base station sends commands or queries to the sensor nodes, and receives reports from them. All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

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 described a military scenario.

In this paper, we present three authentication schemes for addressing this form of attack, which we call a *false data injection attack*. Our schemes enable 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.

More specifically, in our proposed authentication schemes $t+1$ sensor nodes first agree upon a report before it is sent to the base station, where t is a security threshold based on the network node density and the security requirements of the application under consideration. Then, all the nodes that are involved in relaying the report to the base station authenticate the report in *an interleaved, hop-by-hop* fashion. 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 schemes provide an upper bound B for the number of hops that a false data packet can be forwarded before it is detected and dropped. The en-route filtering capability of these three schemes differs as described below:

—In Scheme I, if a node on the path between the base station and a cluster head has the authenticated path knowledge or if t compromised nodes all belong to a cluster, $B = t$. For applications that cannot meet either of the above conditions, in the worst case $B = O(t^3)$.

—Scheme II always guarantees that $B = 0$, i.e., a false report will be dropped immediately. This scheme however has a larger computational cost than Scheme I.

—Scheme III has the merits of the other two schemes. It provides B close to 0 and has the least computational overhead, at the expense of slightly higher additional storage overhead.

In addition to analyzing the security and the performance of each of these schemes, we have also implemented a prototype of Scheme II on MICA2 motes [xbo 2005], which are representative of the current generation of sensor nodes. Our security and performance analysis and prototype implementation indicate that our schemes are effective, efficient (with respect to the achieved security), and also practical.

The remainder of this paper is organized as follows. Section 2 first describes the network, node, and security assumptions, then states our design goals and attack models. Section 3 presents our basic scheme, followed by two improved schemes in Section 4, and Section 5 respectively. For each scheme, we provide a detailed performance and security analysis. We further compare all the proposed schemes in Section 6. We describe our prototype implementation of Scheme II in Section 7. We discuss related work in Section 8, and finally conclude our work in Section 9.

2. ASSUMPTIONS AND DESIGN GOALS

2.1 Assumptions

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

2.1.1 *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 [Hill et al. 2000]) in their computational and communication capabilities and power resources. We assume that every node has space to store several hundred bytes of keying materials.

2.1.2 *Security Assumptions.* We assume that every node shares a master secret key with the base station. We also assume that every node knows the authenticated set of its one-hop neighbors and has established a pairwise key with each of them. For example, we can use the pairwise key establishment scheme in LEAP [Zhu et al. 2003] to achieve this goal. Under this assumption, the impact of a node compromise is *localized* in the immediate neighborhood of the compromised node. That is, an attacker can only inject false packets into the sensor network through the compromised nodes. 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 [Blom 1985] or the Blundo scheme [Blundo et al. 1993] directly. For a larger network, we may use the extensions [Du et al. 2003; Liu and Ning 2003b] 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 [Du et al. 2003; Liu and Ning 2003b]. 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 its broadcast messages (e.g., based on μTESLA [Perrig et al. 2001]), and every node can verify the broadcast messages. As such, we can prevent malicious triggering of broadcast storms. 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 knows all

the keying material of the compromised nodes, and he 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

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 [Goldreich et al. 1986].
—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.

3. SCHEME I: THE BASIC SCHEME

We first give an overview of the basic scheme before discussing it in greater detail. We then discuss issues related to node failure and path dynamics. Finally, we evaluate both the security and the performance of the scheme.

3.1 Definition

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 association node *of node u_j, and u_j is the* lower association node *of node u_i.*

For simplicity, we refer to upper association node and lower association node as UA node and LA node, respectively. From the definition, we know that a node that is less than $t + 1$ hops away from BS has no UA node and an intermediate node may have multiple LA nodes if it has multiple child nodes leading to multiple

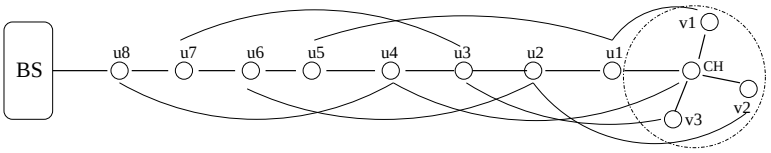

Fig. 2. An example showing the definition of *association* where $t = 3$. BS is the base station and CH is a cluster head. Two nodes connected with an arc are associated, the one closer to the base station is the *upper* association node and the other is the *lower* association node.

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 LA 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 BS, where $t = 3$. Node u_3 has an UA node u_7 and a LA node v_3. Node u_5 has a LA node u_1 but no UA node.

3.2 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 BS 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 UA node. Then it sends the MACs to its cluster head. The cluster head CH 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 LA node, and then removes that MAC from the received report. If the verification fails, it drops the report. Otherwise, it then computes and attaches a new MAC based on its pairwise key shared with its UA 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 discards the report.

3.3 Scheme Description

This subsection describes the basic processes in the scheme. Subsections 3.4 and 3.5.2 discuss some of the issues in more detail.

3.3.1 *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 pairwise key establishment scheme in LEAP [Zhu et al. 2003] is employed for establishing one-hop pairwise key, node u will be loaded with an initial network key. If the Blundo scheme [Blundo et al. 1993] 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.

3.3.2 *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 processes – *BS Hello* and *CH Acknowledgment*. We then describe an incremental association discovery scheme in Section 3.4, which is executed when the upper and/or lower association nodes of a node change due to the change of the path from the CH to the BS.

3.3.2.1 *BS Hello.* This process enables a node to discover its UA node. The base station BS initiates this process by broadcasting a HELLO message, which is recursively forwarded to all nodes so that every node discovers the ids of up to $t+1$ closest nodes that are on its path to the BS. 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 HELLO message to $t+1$. Specifically, 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, irrespective of the number of hops the HELLO message travels. On receiving the HELLO message, the cluster head CH 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. Note that here an authentication scheme such as μTELSA is used to provide authenticity of the original BS HELLO message. However, it can only prevent malicious broadcast storms, but cannot ensure authenticity of the modified part of the HELLO message.

Fig. 3 shows an example where $t = 3$. The cluster consists 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 and message loops. M is authenticated by an authentication scheme such as μTELSA. Note that here μTESLA prevents malicious broadcast storms. For

After receiving M, node u_6 records the id(s) in M, attaches its own id to M,

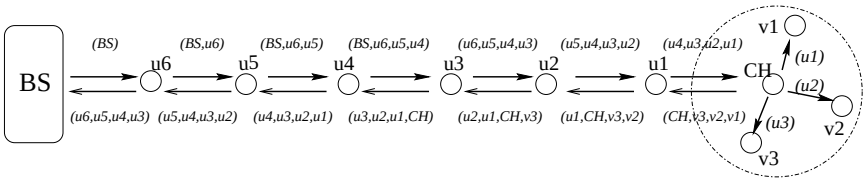

Fig. 3. An example illustrating the BS hello process where $t = 3$. BS is the base station, u_i is an en-route node. CH is the cluster head and v_1, v_2, v_3 are cluster nodes. (M) is the content of the beaconing message. Note that u_i may be an en-route node for multiple paths and CH may also be an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

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 UA node.

3.3.2.2 *CH Acknowledgment.* After the BS hello process, the cluster head CH sends an acknowledgment back to the BS. The acknowledgement includes a cluster id, and the ids of the $t + 1$ LA nodes. When a node receives the acknowledgement, it will check if all the node ids in the message are distinct. If not, it will drop the message because that indicates an attack. During the forwarding of the acknowledgement, the node ids are replaced in the direction opposite to that in the BS hello process, that is, a node removes the last id in the id list and adds its own id at the beginning. This allows every receiving node to discover the id of its LA node. In the case that a node has multiple child nodes leading to multiple clusters, the node has multiple LA nodes. Therefore, it maintains a table that includes multiple path information, where each path is uniquely identified by the corresponding cluster id. Moreover, because the CH acknowledgment message is critical for a node to maintain correct association knowledge, we employ a hop-by-hop acknowledgment mechanism to avoid packet loss due to unreliable link layer transmission.

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 UA nodes in the BS hello process. CH sends the acknowledgment to u_1, which forwards the HELLO message to CH in the BS Hello process. The id list in the acknowledgment message is $\{CH, v_3, v_2, v_1\}$. Based on the id placement rule, u_1 discovers its LA node 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 $\{u_1, CH, v_3, v_2\}$. In this way, every node on the path finds out its LA node, while the size of the acknowledgment message

remains bounded.

During this process, 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 message to its upstream 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. The security of this process is analyzed in Section 3.6.

3.3.3 *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 or an incremental sequence number, and the cluster id C_i), it computes a MAC over E, using its authentication key K_v^a as the MAC key. In addition, node v computes another MAC over E, using the pairwise key shared with its UA 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 establishment scheme because they know each other's id from the association discovery phase. We refer to these two types of MACs as *individual MAC* (IMAC for short) and *pairwise MAC* (PMAC for short), respectively. Node v then sends an endorsement message to the CH that includes these two MACs. The CH collects endorsements from $t+1$ cluster nodes (including itself). It then compresses the $t+1$ IMACs by XORing them to reduce the size of a report. The PMACs however are not compressed for transmission, because otherwise a node relaying the message will not be able to extract the PMAC from its LA node. The CH finally generates a report, which contains the event E, a list of ids of the endorsing nodes, the compressed MAC and $t+1$ PMACs. We will discuss the use of a short PMAC to reduce the message overhead in Section 3.7.

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 $K_{v_1}^a$ 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 $XMAC(E)$.

$$\begin{aligned} XMAC(E) \; = & MAC(K_{v_1}^a, E) \oplus MAC(K_{v_2}^a, E) \oplus \\ & MAC(K_{v_3}^a, E) \oplus MAC(K_{CH}^a, E). \end{aligned}$$

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

$$\begin{aligned} R: \quad & E, \{v_1, v_2, v_3, CH\}, XMAC(E), \\ & \{MAC(K_{CHu_4}, E), MAC(K_{v_3u_3}, E), \end{aligned}$$

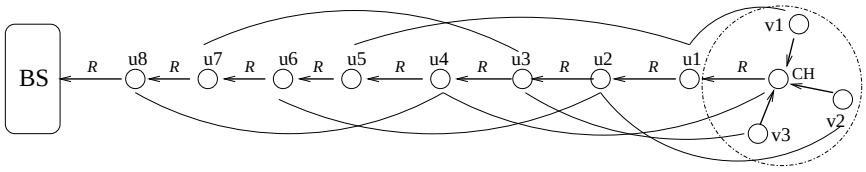

Fig. 4. An example of report endorsement and en-route filtering where $t = 3$.

$$MAC(K_{v_2u_2}, E), MAC(K_{v_1u_1}, E) \oplus MAC(K_{CHu_1}, E)\}.$$

The report includes the ids of the endorsing nodes v_1, v_2, v_3 and CH, which enable 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 contained in E. The order of the PMACs in R corresponds to that in the CH acknowledgment message so that a node receiving R knows which PMAC is from its LA node. Note that here the last PMAC is actually the result of an XOR operation of two MACs, the first contributed by v_1 for its UA node u_1 and the second contributed by CH for its immediate upstream node u_1. The second MAC is important for restricting the impact of a node compromise to its direct neighbors. If a report R does not include this MAC an attacker can inject false data packets at any location of a sensor network once it has compromised a single node. Since each report includes this MAC, an attacker that has compromised t nodes can only inject false data into a network through these t nodes. This is because each node is assumed to have obtained authenticated neighbor knowledge using the pairwise key establishment scheme in LEAP [Zhu et al. 2003]. The reason for piggybacking this MAC on a PMAC using an XOR operation is to reduce message overhead without reducing security.

3.3.4 *En-route Filtering.* When a node u receives R from its downstream node, it first checks the number of different PMACs in R. If node u is $s(s < t + 1)$ hops away from BS, it should see s PMACs; otherwise, it should see $t + 1$ PMACs. It then verifies the last PMAC in the PMAC list, based on its pairwise key shared with its LA node and its pairwise key shared with its immediate downstream neighbor. In the case that it has not computed the pairwise keys earlier, it computes the pairwise keys and stores it. Node u will drop the report if the above checks fail. Otherwise, if node u is more than $t+1$ hops away from BS, it proceeds to compute a new PMAC over event E using the pairwise key shared with its own UA node and its pairwise key shared with its immediate upstream node. It then removes the last PMAC from the PMAC list and inserts the new PMAC into the beginning of the PMAC 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 four PMACs. If true, it computes its pairwise key shared with node v_1, i.e., $K_{u_1v_1}$, if it has not computed the key before. Node u_1 then verifies the last PMAC in R by computing $MAC(K_{v_1u_1}, E)$ and $MAC(K_{CHu_1}, E)$ and then XORing them. If the verification succeeds, node u_1 computes $MAC(K_{u_1u_5}, E)$, i.e., the new PMAC computed over E using the pairwise key shared with node

u_5. It also computes $MAC(K_{u_1u_2}, E)$ for its upstream node u_2. Finally, node u_1 inserts the $MAC(K_{u_1u_5}, E)$ at the beginning of the PMAC list, and removes the last PMAC on the list. The report R that node u_1 forwards to node u_2 is as follows.

$$\begin{aligned} R: \quad & E, \{v_1, v_2, v_3, CH\}, XMAC(E), \\ & \{MAC(K_{u_1u_5}, E), MAC(K_{CHu_4}, E), \\ & MAC(K_{v_3u_3}, E), MAC(K_{v_2u_2}, E) \oplus MAC(K_{u_1u_5}, E)\}. \end{aligned}$$

The other forwarding nodes follow the same process, except that the nodes within $t+1$ hops of BS do not insert a new PMAC. It is easy to see that every node on the path from the cluster head to the base station can verify one PMAC in the report independently. Thus the report is authenticated in an interleaved hop-by-hop fashion.

3.3.5 *Base Station Verification.* The BS only needs to verify the compressed MAC. Basically, it computes $t+1$ IMACs over E using the authentication keys of the nodes in the id list. It then XORs the MACs to see if the resulting XMAC 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, the BS can react to the event; otherwise, it will discard the report.

3.4 Association Maintenance

The correctness of our scheme relies on correct association knowledge. A node needs to know the id of its LA node; otherwise, it will not know which pairwise key to use to verify a PMAC. In addition, it needs to know the id of its UA node so that it can add a valid PMAC into a report; otherwise, its UA 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*.

3.4.1 *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 [Hill et al. 2000], 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 received 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 BS hello process for each epoch by piggybacking node ids in every beaconing message. The CH acknowledgment process can be omitted by letting a LA node include its id with its PMAC when it forwards a report. This strategy works best for networks where the topology changes very frequently, at the additional bandwidth expense of including $t+1$ ids per beaconing message. For less dynamic networks, this overhead will be re-

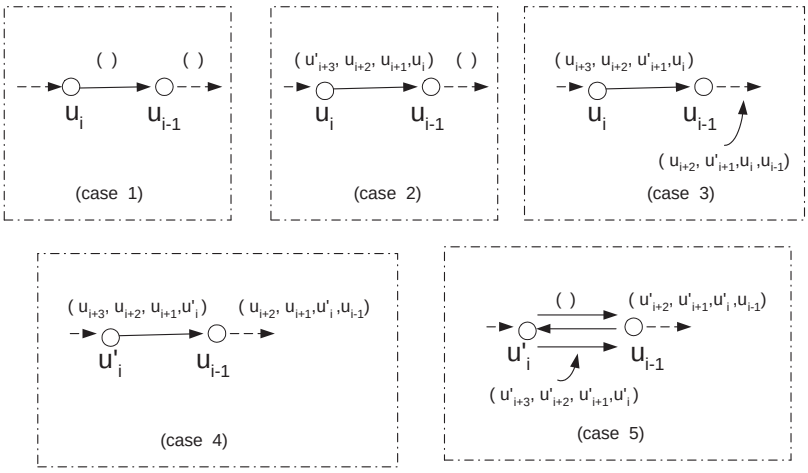

Fig. 5. Five basic cases for base station initialized repair when $t = 3$. In case 1, node u_{i-1} does not add its id into a beaconing message when it finds there is no ids in the received message and the message is from its old upstream node u_i. In case 2, node u_{i-1} finds only its own UA node has changed, it restores the beaconing message to its original format (i.e.,no piggybacked ids). In case 3 and case 4, whether its upstream node u_i is the same or not, node u_{i-1} attaches its own id based on the id replacement rule. In case 5, u_{i-1} finds its upstream node has changed and the beaconing message comes with no ids, it requests the id list and then sends the replaced list to its downstream node.

duced. If a path does not change during different epochs, it is not necessary for a node to attach its id to a beaconing message.

We adopt a reactive approach for association maintenance in relatively static networks. Recall that in the BS hello process, 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. Figure 5 illustrates five basic scenarios and the way they should be handled by node u_{i-1}.

3.4.2 *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, however, will result in inconsistent node association relationship in our scheme. Thus, we need an adaptive scheme to locally repair the association relationship.

Our general idea is as follows. The immediate downstream node of a failed node will send a LA repair message to an upstream node based on an existing route repair protocol. This LA repair message includes two id lists: the id list of the $t+1$ downstream nodes of the failed node and the id list of the t upstream nodes of the same failed node. The first list helps the new upstream nodes find their new LA nodes when the message is forwarded towards BS and modified hop-by-hop like an CH acknowledgement message; the second list helps merge the new path into the old path to minimize the path rebuilding cost. The LA repair message is forwarded until it reaches one of the t upstream nodes of the failed node or an upstream node whose all $t+1$ downstream nodes are the same as before. This upstream node will send downwards a UA repair message, which contains the ids of its t upstream nodes. The UA repair message is forwarded and modified hop-by-hop like a BS HELLO message until it reaches the LA node of the failed node. In this way, all the affected nodes will have the updated association information, which allows these nodes to compute their pairwise keys with the newly associated nodes.

Consider an example in Figure 6 where node u_5 has failed and here $t = 3$. u_4 detects the failure of its upstream node (the issue of node failure detection is out of our scope) and tries to find a path towards the base station. If w_1, w_2, w_3 previously did not belong to a path, we can apply the right-hand rule in the greedy parameter stateless routing (GPSR) protocol [Karp and Kung 2000] for local repair. Here we assume every node knows the locations or relative locations of its neighbors. u_4 sends a LA repair message to w_1, which is the first node counterclockwise about u_4 from edge (u_4, u_5). The LA repair message includes the ids of $t + 1 = 4$ LA nodes of u_5 and $t = 3$ upstream nodes of u_5; that is, it includes two id lists $\{u_4, u_3, u_2, u_1\}$ and $\{u_6, u_7, u_8\}$. Thus, w_1 knows its new LA node is u_1. Since w_1 is not on the old path and its LA node is changed, it will change the first id list in the message into $\{w_1, u_4, u_3, u_2\}$ and forward the message to w_2. Following the same process, w_2 and w_3 forward the repair message based on the same right-hand rule, and they will modify the first id list based on the id placement rule. When u_6 receives the message, it finds that it is in the second list $\{u_6, u_7, u_8\}$, which indicates that the failed node u_5 has been bypassed. u_6 then sends a UA repair message to w_3, which includes the id list $\{u_6, u_7, u_8, u_9\}$. Node w_3 then knows its UA node is u_9. It forwards the list to its downstream nodes in the same way as in the BS hello process, thus u_1 will discover its new UA node w_1. On the other hand, because the LA nodes of u_7, u_8, u_9 have changed, u_6 further forwards the LA repair message towards u_9.

Figure 6 does not include the scenario where the new path does not merge into the old one according to the right-hand rule. However, our general idea still works. Also note that although local repair is necessary to maintain path connectivity,

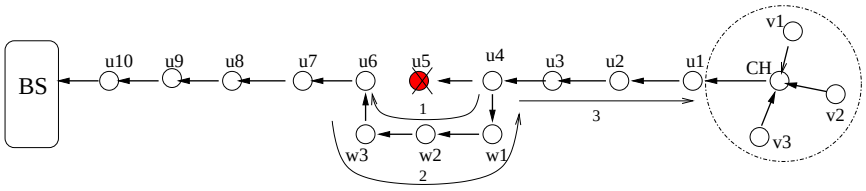

Fig. 6. An example showing the local repair process when node u_5 fails and node u_4 establishes a new path towards BS (here $t = 3$). The new path includes nodes w_1, w_2 and w_3.

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 local repair operations to be at most one within one beaconing epoch.

3.5 Further Discussions

Next we discuss two slight variants of this scheme to copy with more complex situations.

3.5.1 *Dealing with Dynamic Clusters.* In the description of our basic scheme, we have implicitly assumed that when an event is detected, the cluster membership formed in the previous CH acknowledgement process has not changed. Otherwise, not knowing their UA nodes, the new cluster nodes cannot generate correct PMACs. Next we show a slight variant of the basic scheme to handle the dynamic cluster case where one or multiple of the cluster nodes that will contribute endorsements are different from those participating in the previous CH acknowledgement process.

There are two strategies to consider. First, if the cluster nodes or the cluster head hold the event data until the next BS HELLO comes, the average delay will be half the BS hello interval. Second, when the cluster head notices the change of cluster membership since the previous CH acknowledgement, it can associate the new cluster nodes with appropriate en-route nodes. In this way, the new cluster nodes can know their UA nodes and generate correct PMACs. The cluster node then directly forwards the endorsed data towards the BS. Slightly different from the basic scheme, in this case the report will also carry the ids of the new cluster nodes so that the UA nodes of these new cluster nodes may verify the PMACs (other en-route nodes will not be affected). Note that the security strength of this variant is the same as the basic scheme because a BS hello process is not protected from insider attacks anyway.

3.5.2 *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 when it is beneficial. As described earlier, we can integrate the BS hello process

with the TinyOS beaconing protocol [Hill et al. 2000] 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 [Karp and Kung 2000] 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.

3.6 Security Analysis

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

3.6.1 *Base Station Detection.* Our authentication scheme requires that each of $t+1$ cluster nodes compute an IMAC 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 $t+1$ IMACs into one XMAC based on the bitwise XOR operation (instead of attaching $t+1$ IMACs) to reduce message overhead. This compression scheme is secure because it is a special case of the XOR-MAC scheme [Bellare et al. 1995] which is proven to be secure.

3.6.2 *En-route Filtering.* Next 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.

3.6.2.1 *Outsider Attacks.* An outside attacker could eavesdrop, replay, or inject messages. Eavesdropping does not hurt the filtering capability of enroute nodes. In our scheme, every report is authenticated at every hop during its transmission. Thus, any false data injected by an outsider will be detected and dropped immediately. Moreover, because an event also contains a timestamp or a sequence number, an attack in which an outsider replays an old report will be detected.

3.6.2.2 *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 UA node and its LA node. This corresponds to a situation in which every node is loaded with correct association knowledge before it is deployed, or every node discovers the ids of its association nodes through the association discovery process before any nodes are compromised.

LEMMA 1. *Given authenticated association knowledge, a false report injected by t compromised nodes is dropped after it is forwarded by at most t noncompromised nodes.*

PROOF. A compromised node can provide an authenticated PMAC over any data to deceive its UA node. Thus, if totally t nodes are compromised, they can provide t authenticated PMACs over a false report, which will pass the verification of t noncompromised UA nodes. By requiring that every en-route node verify a PMAC from its LA node, our scheme enables one noncompromised node among any $t+1$ consecutive en-route nodes to filter out a false report because it will not receive

a valid PMAC from its LA node. Note that in a special case when the distance between a CH and the BS is less than $t+1$ hops, the BS will discard a false report based on $XMAC$. Thus, despite the distance between a CH and the BS, our scheme guarantees that a false report will be dropped after it is forwarded by at most t noncompromised nodes. □

Next we analyze the security of our scheme in another scenario where no authenticated association knowledge is provided. Essentially, we need to analyze the security of the association discovery process because it provides association knowledge to nodes. More specifically, the security of the *cluster acknowledgment* process is critical because it provides the nodes with the lower association knowledge that is used as the basis for en-route filtering. The cluster acknowledgment process is subject to attack if it is executed at any time after some 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, the node will set the id of its LA 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 (or less) compromised nodes*, under the constraint that $t+1$ distinct ids must appear in the list when the ACK is forwarded. This attack is referred to as *mis-association attack*. 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.

Below we discuss two types of insider attacks on en-route filtering: *cluster insider attack* and *en-route insider attack*. The strength of other insider attacks falls between.

Cluster Insider Attack 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.

LEMMA 2. *In a cluster insider attack, a false report is dropped after it is forwarded by at most t noncompromised nodes.*

PROOF. 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 nonexistent node. Therefore, one of the $t+1$ consecutive 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 nonexistent node. Since a noncompromised or nonexistent node does not contribute a valid MAC over the forged event, the attacker has to forge a MAC corresponding to the forged node id, thus the en-route node lower associated with the forged node will detect the forged MAC and drop the false report. Hence, a false report will be dropped after it is forwarded by at most t

noncompromised nodes. □

En-route Insider Attack In this attack, t compromised nodes that lie on the path to the base station collude to attack the cluster acknowledgment process. If $t = 1$, a false report will be dropped immediately. Although an attacker can lower associate an en-route node to two nodes of its choice, it cannot provide two valid MACs in a false report to an en-route node later because an en-route node verifies two MACs. Therefore, we assume $t > 1$ in the following discussion. We first construct an attack to show that in the basic scheme without authenticated association knowledge the en-route nodes cannot filter out the false reports injected by t colluding nodes, then give a related upper bound.

From the previous security analysis of our basic scheme, we can see that if an attacker can deceive $t+1$ en-route nodes in a row, the basic scheme will not provide any en-route filtering capability, because the next $t + 1$ nodes in a row trust the previous $t + 1$ nodes. Hence, given the assumption that an attacker can only compromise t nodes, the goal of an attacker is to deceive $t + 1$ noncompromised nodes through these t compromised sensors. An attacker may achieve this goal by associating multiple upstream nodes to one compromised downstream node, if we assume that an attacker can selectively compromise en-route nodes and the en-route nodes do not have any authenticated association knowledge.

Here is a specific attack. Denote the t compromised nodes on the path as $X_1, X_2, ..., X_t$, where X_i is farther away from BS than X_j is if $i < j$. Let the number of noncompromised nodes between X_1 and X_2 be m ($m \leq t$). Node X_1 can choose m ids of compromised nodes and then arrange them in its CH acknowledgement message such that m noncompromised nodes will be lower associated with these m compromised nodes. Not knowing that they have been deceived, these m noncompromised nodes will then provide m authenticated PMACs to their UA nodes. Now in the CH acknowledgement message, X_2 could forge an id list containing $t + 1$ ids chosen randomly from the ids of the previously deceived m nodes and t compromised nodes. The next $t + 1$ nodes in a row will be lower associated with these $t + 1$ nodes that will later provide $t + 1$ authenticated PMACs. Thus, a false data packet will not be detected by en-route nodes.

Next we derive an upper bound on the number of hops a false report can traverse. Let the maximum number of node ids in a BS hello message and a CH acknowledgement message be r (instead of $t + 1$ in the basic scheme).

LEMMA 3. *In an en-route insider attack where an attacker can selectively compromise en-route nodes and the en-route nodes do not have any authenticated association knowledge, if $r \geq t^2 + 1$ and every node accepts an id list only when all the ids are distinct, at most $B_{max} = \frac{t^2(t+1)}{2}$ noncompromised nodes will be deceived to forward false data packets injected by t colluding compromised nodes.*

PROOF. We first construct a specific attack scenario in which the number of nodes deceived to forward false data is B_{max}, and then show B_{max} is also the upper bound for more general attack scenarios. Again, denote the t compromised nodes on the path as $X_1, X_2, ..., X_t$, where X_i is farther away from BS than X_j is if $i < j$. In this specific attack, t nodes are selectively comprised such that the

number of noncompromised nodes between X_i and X_{i+1} is $t \cdot i$. This is illustrated as follows:

$$\begin{aligned}
&X_1, \{u_{1,1}, u_{1,2}, ..., u_{1,t}\},\\
&X_2, \{u_{2,1}, u_{2,2}, ..., u_{2,2t}\},\\
&...,\\
&X_{t-1}, \{u_{t-1,1}, u_{t-1,2}, ..., u_{t-1,(t-1)t}\},\\
&X_t, \{u_{t,1}, u_{t,2}, ..., u_{t,t^2}\},\\
&..., BS
\end{aligned} \tag{1}$$

where $u_{i,j}$ is a noncompromised node.

Node X_1 first forges a list of r ids, among which the first $r-t$ are randomly picked legitimate ids and the remaining t ids are those of the t compromised nodes. For example, the id list could be $\{Y_1, ..., Y_{r-t}, X_1, X_2, ..., X_{t-1}, X_t\}$. According to the id replacement rule in the cluster acknowledgment process, every noncompromised node between $u_{1,1}$ and $u_{1,t}$ sets its lower association node to be the last id in the list, removes the last id and then inserts its own id at the beginning of the list. As a result, each of the nodes $u_{1,1}, u_{1,2}, ..., u_{1,t}$ is lower associated to one compromised node. Note that the node after $u_{1,t}$ (i.e., X_2) is lower associated with Y_{r-t} according to the id replacement rule. If X_2 were not a compromised node, it would drop any false data packets because it would not receive a valid PMAC from node Y_{r-t}. Since X_2 is a compromised node, it ignores the association rule. This indicates that an attacker cannot deceive more than t noncompromised nodes if X_2 is farther away from X_1. On the other hand, if X_2 is closer to X_1, the attacker can only deceive less than t noncompromised nodes. Thus, in the en-route filtering phase, node X_1 can deceive at most t noncompromised nodes to forward false data.

Next node X_2 forges a new list $\{Y_1, ..., Y_{r-2t}, X_1, X_2, ..., X_{t-1}, X_t, u_{1,t}, u_{1,t-1}, ..., u_{1,1}\}$ and sends it to $u_{2,1}$. Thus $u_{2,1}, ..., u_{2,t}$ are lower associated with $u_{1,1}, ..., u_{1,t}$ respectively, and $u_{2,t+1}, ..., u_{2,2t}$ are lower associated with $X_t, ..., X_1$ respectively. On the other hand, X_2 could have easily forged a list that upper-associates $u_{1,1}, u_{1,2}, ..., u_{1,t}$ to $u_{2,1}, u_{2,2}, ..., u_{2,t}$ respectively in the base station hello phase. Thus, here at most $2t$ noncompromised nodes will be deceived to forward false data. More generally, $X_i (i > 2)$ can forge an id list that lower-associates $t \cdot i$ noncompromised nodes to t compromised nodes and $(i-1)t$ noncompromised nodes that were deceived by X_{i-1}. Thus, the total number of noncompromised nodes that will be deceived to forward false data packets is at most $B_{max} = t + 2t + ... + t^2 = \frac{t^2(t+1)}{2}$.

Note that for the above result to hold, we must set $r \geq t^2 + 1$. This is because X_t can forge an id list to deceive up to t^2 noncompromised nodes. If $r \leq t^2$, the nodes between u_{t,t^2} and BS will not be able to detect a false data packet because each of them will receive a valid PMAC from its claimed LA node.

We derived B_{max} from the above specific attack scenario in which an attacker started from X_1 and ended at X_t in a specific order. Actually, the order is irrelevant because the maximum number of noncompromised nodes deceived by a compromised node always increases according to the sequence $t, 2t, ..., t^2$. Moreover, as shown earlier, changing the number of noncompromised nodes between two compromised nodes does not help deceive more noncompromised nodes either. □

In fact, it is easy to see the total number of deceived noncompromised nodes is upper bounded by B_{max} irrespective of the locations of the t compromised nodes in the network. On the other hand, because a node only accepts reports authenticated by one of its immediate neighbors, an attacker can inject a false report only through a compromised node. More generally, let t_1 and t_2 be the number of compromised nodes in a cluster (including CH) and on the path, respectively, and $t_1 + t_2 = t$. Then when $r \geq t \cdot (t_2 + 1) + 1$, $B_{max} = t_1 + \frac{t_2^2(t_2+1)}{2}$.

3.6.3 *Localizing False Data Injection Capability.* 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 avoid 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 authenticated set of its direct neighbors and establishes a pairwise key with each of them, and it will *only* accept events authenticated by one of the nodes from its neighbor set, which is its current immediate downstream node. This implies that a compromised node can only mount an attack *locally* and *on its own behalf*. Thus, after compromising t nodes, an attacker can only inject false data reports through these t nodes, but not through any noncompromised nodes. In other words, although our scheme has a security threshold of t, it does not mean our scheme is completely broken after more than t nodes are compromised.

3.7 Performance Evaluation

This section analyzes the computational and communication overheads of our basic scheme.

3.7.1 *Computational Cost.* The computational overhead of our scheme is mostly due to two operations – establishing pairwise keys and report authentication.

3.7.1.1 *Establishing Pairwise Keys.* In the basic scheme, two associated nodes need to establish a multi-hop pairwise key on the fly, based on one of the id-based schemes [Blom 1985; Blundo et al. 1993; Du et al. 2003; Liu and Ning 2003a; 2003b]. 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 $65,536$ sensor nodes in a network). The cost of computing a pairwise key has been shown [Liu et al. 2005] to be of the same order as that of generating a MAC based on RC5 [Rivest 1994], which is generally believed to be efficient for sensor networks. Moreover, in our scheme normally a cluster node computes one pairwise key and an en-route node computes two. In the case of a node failure or

a path change, a node computes a pairwise key shared with a new node; however, this situation does not happen very frequently.

3.7.1.2 *Report Authentication.* In the basic 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 PMAC from its LA node and generates one PMAC for its UA node if it is more than t hops away from BS; it verifies one MAC from its downstream neighbor and generates one MAC for its upstream neighbor. All these PMACs are computed over an event E, not the entire report R. Therefore, if we use 4 bytes for an event and the report R is 24 bytes, the computational overhead of computing a PMAC is multiple times smaller than the overhead of computing an IMAC. Note that in our scheme although a forwarding node computes two more MACs than that in a hop-by-hop authentication scheme [Zhu et al. 2003], the security it achieves is much stronger. Since the energy for computing a MAC is about the same as that for transmitting one byte [Ye et al. 2004], 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. Also note that the energy for computing two MACs using the same key is usually much smaller than twice of that for computing one MAC. For example, when computing a MAC based on RC5 [Rivest 1994], the most expensive operation is the initial key setup that outputs a cipher context, which only has to be performed once for a specific key. After this operation, the future computation for generating MACs consumes much smaller energy by reusing the same cipher context. Therefore, as long as a node keeps the cipher context in its memory (about 80 bytes), the computation of MACs will be very efficient.

3.7.2 *Communication Cost.* We consider the communication cost (overhead) of our scheme as the *additional* bandwidth overhead incurred in providing data authentication, compared to a nonsecure packet forwarding protocol which merely delivers the event report through a path. As such, the communication overhead of our scheme comes from the following sources. First, every authenticated report contains one compressed MAC and $t + 1$ PMACs. In practice, we can choose a larger size for an IMAC, while selecting a smaller size for a PMAC. 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 PMAC 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 totally 4 bytes for PMACs, and $t = 3$, the size of a PMAC will be 1 byte. In this case, an attacker, after compromising 3 nodes, has the probability of 1/256 to blindly forge a packet that will pass the en-route filtering process. Second, additional bandwidth overhead is involved during association discovery. In the BS hello process, a node adds its own id to a beaconing message, thus the cost is at most $t+1$ node ids; in the CH acknowledgement process, the cost is at most $t + 2$ node ids and one MAC. We note that there is also communication cost in a local repair process when a path changes; however, this component of the communication overhead depends on path dynamics. For a relatively static

network, this cost could be very small when amortized to all the nodes on a path.

4. SCHEME II: TRADING COMPUTATIONAL OVERHEAD FOR IMMEDIATE DETECTION

In scheme I, the base station can detect an injected false data by t compromised nodes immediately whereas an en-route node can be easily deceived by colluding nodes. Their capability in filtering false data is determined by the number of MACs they verify. This observation motivates the design of Scheme II.

The basic idea underlying this scheme is that every node en-route to the base station accepts a report received from a downstream node only if it has been verifiably endorsed by at least $t+1$ nodes, just as the base station does. To realize this idea, however, we cannot assume that each node en-route to the base station shares an exclusive key with the $t+1$ cluster nodes that originate data reports. Instead, this scheme requires every en-route node to have established pairwise shared keys with $t+1$ nodes that are immediately downstream from it. A report is accepted by the node if it has been endorsed by these associated nodes. As in Scheme I, we first define the node association relationship more formally.

DEFINITION 2. *For two nodes u_i and u_j on the path from CH to BS, if $|i-j| \leq t+1$, we say u_i and u_j are* range associated*, and u_i is a range association node of u_j. More specifically, if $0 < i-j \leq t+1$, u_i is the* upper range association (URA) node*of node u_j, and u_j is the* lower range association (LRA) node *of node u_i.*

From this definition, we know that a node u_i has $t+1$ URA nodes if it is at least $t+1$ hops away from BS, and it has $n-i$ URA nodes otherwise. A node u_i has $t+1$ LRA nodes if it is at least $t+1$ hops away from CH. 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 is also lower range associated with CH and $t+1-i$ cluster nodes v_j $(1 \leq j \leq t+1-i)$.

—The cluster head and its cluster nodes do not have LRA nodes.

DEFINITION 3. *We call the set composed of all the URA nodes of node u the* URA set *of node u, denoted as $U(u)$, and the set composed of all the LRA nodes of node u the LRA set of node u, denoted as $L(u)$.*

Fig. 7 shows a node cluster and a path from the cluster head to the base station, where $t=3$. The LRA set $L(u_4)$ for node u_4 is $\{u_3, u_2, u_1, CH\}$ and its URA set $U(u_4)$ is $\{u_5, u_6, u_7, BS\}$. The URA set of node u_6 (not shown in the figure) is $U(u_6) = \{u_7, BS\}$. Note that an intermediate node may have multiple LRA sets if any of its t downstream nodes has multiple child nodes leading to multiple clusters.

Scheme II also has four phases, as in Scheme I. Below we introduce the scheme in more detail, focusing on its differences from Scheme I.

4.1 Association Discovery

A node can discover its URA and LRA sets through a process similar to the *association discovery* process in Scheme I. The differences are listed below:

—In the BS Hello process, an en-route node records all the ids in the id list as its URA nodes. Upon receiving the HELLO message, the cluster head CH assigns

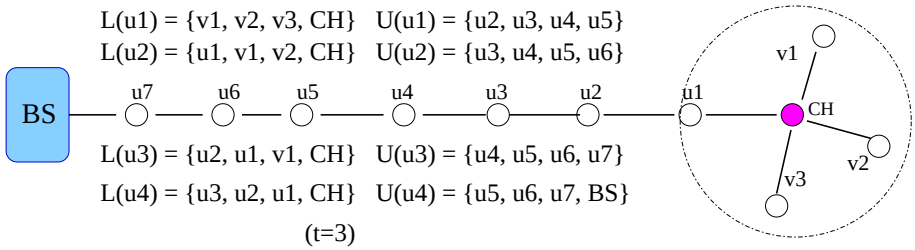

Fig. 7. A logical view of a sensor network in which nodes v_1, v_2, v_3 and CH have been deployed in a cluster to monitor an area of interest. The upper and lower range association sets of nodes u_1, u_2, u_3, and u_4 are shown for $t = 3$, where t is the number of compromised nodes that can be tolerated.

different numbers of ids in the HELLO message to its cluster nodes (including itself). For example, referring to Figure 3, in Scheme I when CH receives M, it assigns one id to each cluster node. In scheme II, CH assigns $\{u_1\}$ to v_3, $\{u_1, u_2\}$ to v_2 and $\{u_1, u_2, u_3\}$ to v_1, respectively.

—In the CH acknowledgement process, an en-route node records all the ids in the id list as its LRA nodes. For example, the CH acknowledgement message CH sends to u_1 includes the id list $\{CH, v_1, v_2, v_3\}$. u_1 knows its LRA set is $\{CH, v_1, v_2, v_3\}$. It then removes v_3 from the list and inserts its own id at the beginning of the list. The id list it sends to u_2 is then $\{u_1, CH, v_1, v_2\}$.

4.2 Report Endorsement

When a node v agrees on an event E, it computes an IMAC for E, using its authentication key K_v^a as the MAC key. Suppose node v has s URA nodes, discovered through the BS hello process. Node v computes s PMACs over E, each of which is based on the pairwise key shared with each of its s URA nodes. Node v then sends an endorsement message to the cluster head, which includes its IMAC and s PMACs. As in Scheme I, the cluster head collects endorsements from $t+1$ cluster nodes (including itself), and then compresses the $t+1$ IMACs by XORing them to reduce the size of a report. Unlike in Scheme I, the PMACs for every URA node are also compressed based on the XOR operation. These two types of compressed MACs are referred to as XIMAC and XPMAC, respectively. The cluster head finally generates a report, which contains the event E, a list of ids of the endorsing nodes, the XIMAC and $t+1$ XPMACs.

Consider the cluster node v_2 in Fig. 7. v_2 computes three MACs over the event E, one IMAC and two PMACs for its URA nodes u_1 and u_2, respectively. v_2 sends its endorsement that contains these MACs to CH. The endorsement is authenticated with the pairwise key shared between v_2 and CH.

CH collects endorsements from the other two nodes v_1 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 XIMAC over E, denoted as $XIMAC(E)$.

$$XIMAC(E) = MAC(K_{v_1}^a, E) \oplus MAC(K_{v_2}^a, E) \oplus MAC(K_{v_3}^a, E) \oplus MAC(K_{CH}^a, E).$$

CH further composes four compressed XPMACs based on the PMACs contributed by the cluster nodes (including itself).

$$XPMAC_{u_4}(CH) = MAC(K_{u_4CH}, E)$$
$$XPMAC_{u_3}(CH, v_1) = MAC(K_{u_3CH}, E) \oplus MAC(K_{u_3v_1}, E)$$
$$XPMAC_{u_2}(CH, v_1, v_2) = MAC(K_{u_2CH}, E) \oplus MAC(K_{u_2v_1}, E) \oplus MAC(K_{u_2v_2}, E)$$
$$XPMAC_{u_1}(CH, v_1, v_2, v_3) = MAC(K_{u_1CH}, E) \oplus MAC(K_{u_1v_1}, E) \oplus MAC(K_{u_1v_2}, E) \oplus MAC(K_{u_1v_3}, E)$$

The report R that CH finally generates and forwards towards BS is:

$$R: \quad E, \{v_1, v_2, v_3, CH\}, XIMAC(E), XPMAC_{u_4}(CH), XPMAC_{u_3}(CH, v_1), XPMAC_{u_2}(CH, v_1, v_2), XPMAC_{u_1}(CH, v_1, v_2, v_3).$$

The order of the XPMACs in R is such that the next receiving node can verify the last XPMAC.

4.2.1 *En-route Filtering.* An en-route node receiving a report verifies the last XPMAC in the XPMAC list, based on its pairwise keys shared with its LRA nodes. If the verification succeeds, the node u proceeds to compute s_2 PMACs over event E using the pairwise keys shared with its own URA nodes, where s_2 is the number of its URA nodes. It then removes the last XPMAC from the XPMAC list and updates the XPMAC list with these s_2 PMACs. Finally it forwards the report to its upstream node.

Consider node u_1 in Fig. 7. When node u_1 receives the report R from node CH, it checks if there are four XPMACs. It then computes four PMACs over E based on its pairwise keys shared with nodes CH, v_1, v_2, v_3, and derives a XPMAC by XORing these PMACs. Now it can verify the last XPMAC in R, $PMAC_{u_1}(CH, v_1, v_2, v_3)$. If the verification succeeds, node u_1 computes four new PMACs over E, using its pairwise keys shared with its URA nodes u_2, u_3, u_4, u_5. It then removes the last XPMAC in the list. The XPMAC list is updated as follows:

$$XPMAC_{u_5}(u_1) = MAC(K_{u_5u_1}, E)$$
$$XPMAC_{u_4}(u_1, CH) = XPMAC_{u_4}(CH) \oplus MAC(K_{u_4u_1}, E)$$
$$XPMAC_{u_3}(u_1, CH, v_1) = XPMAC_{u_3}(u_1, CH, v_1) \oplus MAC(K_{u_3u_1}, E)$$
$$XPMAC_{u_2}(u_1, CH, v_1, v_2) = XPMAC_{u_2}(CH, v_1, v_2) \oplus MAC(K_{u_2u_1}, E)$$

Finally, the report R that node u_1 forwards to node u_2 is as follows.

$$R: \quad E, \{v_1, v_2, v_3, CH\}, XIMAC(E), XPMAC_{u_5}(u_1), XPMAC_{u_4}(u_1, CH), XPMAC_{u_3}(u_1, CH, v_1), XPMAC_{u_2}(u_1, CH, v_1, v_2).$$

All the other forwarding nodes carry out the same process. However, the nodes within $t + 1$ hops of BS do not insert a new XPMAC. It is very easy to see that every node on the path from the cluster head to the base station can verify one XPMAC in the report independently.

4.2.2 *Base Station Verification.* The base station BS only needs to verify the XIMAC, as in Scheme I. If the report is authenticated, BS then react to the event; otherwise, BS will discard the report.

4.3 Security Analysis

Like in Scheme I, since the base station verifies $t+1$ MACs, it can detect injected false data packets by up to t colluding nodes. Unlike in Scheme I, this scheme enables an en-route node to filter false data injected by t nodes as well because it verifies $t+1$ MACs. No matter how an attacker manipulates the ids in a CH acknowledgement message, as long as an en-route node verifies $t+1$ PMACs by $t+1$ distinct nodes, the attacker cannot forge a valid XPMAC. Therefore, we have the following lemma.

LEMMA 4. *In Scheme II, if an en-route node verifies $t+1$ PMACs provided by $t+1$ distinct nodes, it will immediately detect false data packets injected by t colluding compromised nodes, i.e., $B_{max} = 0$.*

We note that the above claim holds when the size of a PMAC is large enough. In practice, if we use a smaller PMAC (of one byte, for example) an attacker will have a non-negligible success probability in blindly forging data report. This is a tradeoff between security and performance.

4.4 Performance Evaluation

4.4.1 *Computational Cost.* Each node establishes a pairwise key with each of its URA and LRA nodes, if any. Thus, the number of pairwise key establishments is up to $2(t+1)$. Section 3.7.1 showed that the computational overhead of establishing a pairwise key based on Blundo scheme is in the same order of magnitude as that of an AES encryption; therefore, the involved computational cost for computing $2(t+1)$ pairwise keys is affordable. For a sensor network, the value of t is usually not large because of the default small packet size. For example, as we will show in Section 7, the default data payload size is 29 bytes in TinyOS, which limits the value of t to 4.

For every data report, an en-route node computes $t+1$ MACs to verify a XPMAC, and then computes another $t+1$ MACs for its URA nodes if the report is valid. Hence, totally $2(t+1)$ MACs are needed when a node forwards a data report. Given that the energy consumed in computing a MAC is roughly equivalent to that used in transmitting one byte, this is a beneficial trade-off when t is not large.

4.4.2 *Communication Cost.* In Scheme II, both a BS hello message and a CH acknowledgement message contain up to $t+1$ node ids, which are the same as in Scheme I. A report contains one IMAC and $t+1$ XPMACs. Since the size of a XPMAC is the same as that of a PMAC, the per report bandwidth overhead in Scheme II is also the same as in Scheme I.

Scheme I and Scheme II can be thought of as two extremes in a family of protocols based on interleaved hop-by-hop authentication. In both these schemes, an en-route node verifies a MAC from its immediate downstream node with the goal of localizing the impact of a node compromise. In Scheme II, an en-route node also verifies t additional MACs by the other t nodes in its LRA set, whereas in Scheme I an en-route node only verifies one additional MAC by the node that is the farthest to it in its LRA set. This indicates that we can explore the design of schemes whose performance and security properties lie in between those of the two extremes by making a tradeoff between computation and security. A more interesting question

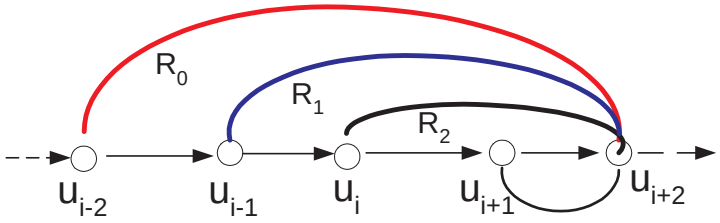

Fig. 8. An example of the hybrid scheme where $t = 3$. Here R_j $(0 \leq j \leq 2)$ is a report whose sequence number equals j (mod 3). The lower association set of node u_{i+2} contains $u_{i-2}, u_{i-1}, u_i, u_{i+1}$. In addition to always verifying a MAC from its immediate downstream node u_{i+1}, for every report, node u_{i+2} also verifies a MAC provided by one of the other three nodes selected in a round robin fashion.

is: can we achieve the best sides of these two schemes? That is, is there a scheme that only requires an en-route node to verify two MACs but provides immediate filtering capability? We introduce such a scheme in the next section.

5. SCHEME III: A HYBRID SCHEME

In Scheme I, for every data report an en-route node only verifies two MACs provided by two predetermined nodes. Thus, an attacker only has to compromise these two fixed nodes to deceive this en-route node. The strategy of Scheme III is to enable a node to verify MACs provided by nodes other than two fixed nodes. Specifically, to localize the impact of a node compromise, for every report a node will always verify a MAC provided by its immediate downstream node in its LRA set. The second MAC of a report could be provided one of the other nodes in its LRA set. To entail the maximum number of distinct nodes in providing MACs, these nodes are required to take turn in providing the second MAC over multiple reports. As such, a node only has to verify two MACs for each report and verifies $2t$ MACs using $t+1$ different pairwise keys shared with its LRA set for every block of t consecutive reports. If an attacker has compromised t en-route nodes, it can generate t valid MACs that may be filled in $t-1$ false reports. Therefore, by letting a node discard the subsequent packets when it fails to receive valid MACs from $t+1$ LRA nodes in t consecutive reports, our scheme guarantees that the number of false reports an attacker can inject in a row is at most $t-1$. Below we describe Scheme III in more detail.

5.1 Description of The Scheme

We first describe the scheme through an example. Figure 8 depicts the node association relationship when $t = 3$. The lower association set of node u_{i+2} contains $u_{i-2}, u_{i-1}, u_i, u_{i+1}$. According to Scheme I, for every report u_{i+2} only verifies two MACs, one from u_{i-2} and the other from u_{i+1}. In Scheme III, however, if the sequence number s of a report satisfies $s \equiv 0$ (mod t), u_{i+2} will verify one MAC from u_{i-2} and the other one from u_{i+1}; if $s \equiv 1$ (mod t), u_{i+2} verifies one MAC from u_{i-1} and the other one from u_{i+1}. More generally, if $s \equiv j$ (mod t), $0 \leq j \leq t-1$,

u_{i+2} will verify one MAC from u_{i+j-2} and the other one from u_{i+1}. Moreover, the sequence numbers of the received reports, when modulo t, must be strictly cyclic between 0 and $t-1$ without any skip. Node u_{i+2} drops any reports whose sequence numbers do not follow the rule. As such, an attacker may inject $t-1=2$ data reports after it has compromised $u_{i-2}, u_{i-1}, u_{i+1}$ in the figure. Node u_{i+2} will drop the following fake reports from the attacker because none of them will include a valid MAC from node u_i. More generally, the number of injected false data packets is bounded by $t-1$ in the case of t compromised nodes.

The enforcement of the above cyclic verification rule poses a strict requirement that data reports must be forwarded reliably in each hop. If a node drops a new report immediately because of the loss of the previous report, in the worst case a node may drop $t-1$ authentic reports even without being attacked. The techniques of hop-by-hop acknowledgement and retransmission could be employed to address the problem, but they do not provide guarantee because in practice the number of retransmissions for a packet is usually limited.

Our solution to avoid new reports being dropped due to packet loss is to let every en-route node maintain a queue of t spaces for accommodating t reports. These t spaces are not necessarily for buffering the most recent t reports a node has received. Instead, a node reserves one space for each of the nodes in its LRA set except the immediate downstream one. For example, in Figure 8, node u_{i+2} has a space for each of u_{i-2}, u_{i-1} and u_i, and each space is used to temporarily buffer the most recent report which could not be forwarded (an older report) or has not been forwarded yet (a new report waiting for its turn to be delivered). If multiple reports authenticated by the same node could not be delivered, the most recent one overwrites the previous ones. More specifically, let i be the sequence number of the last report that was forwarded successfully (e.g., acknowledged) by a node, the next report to be delivered is always the one with the sequence number l such that $(l-i) \equiv 1 \pmod{t}$, irrespective of the arrival times of the reports. If the report l is forwarded successfully, it is removed from the queue. The node then sets $i=l$ and continues the process until no reports are left in the queue. In this way, when a report is lost, the next $t-1$ reports may still be delivered, though with some delay.

5.2 Performance and Security Analysis

The computational overhead of Scheme III is the same as in Scheme I. The security of this scheme however is much stronger than that provided by Scheme I under the same attack because a noncompromised node only forwards up to $t-1$ false reports even when t of its LRA nodes are compromised.

We notice a special *packet substitution attack* may work against Scheme III. To inject false data packets without being detected by an en-route node, an attacker may selectively substitute authenticated packets with false ones. Take the scenario in Figure 8 as an example. Assume that all the LRA nodes of u_{i+2} except u_i have been compromised. An attacker may command a compromised node, e.g., u_{i-2} or u_{i-1}, to substitute two authentic reports that do not require u_i's MAC with two forged data packets, while the authentic packets with u_i's MAC will be forwarded to u_{i+2} without any malicious change. In this way, according to the rule of Scheme III, u_{i+2} will not block the forwarding of the following packets because it can always verify two correct MACs for each packet. Hence, after the third authentic report

Table I. Comparison of Four Schemes ($t > 1$ compromised nodes)

	Security		Performance(#MACs))	
Scheme	BS Detection	En-route Filtering(worst)	Comp.	Comm.
Scheme I	Yes	$O(t^3)$	4	$t+1$
Scheme II	Yes	1	$2(t+1)$	$t+1$
Scheme III	Yes	1	4	$t+1$

passes by node u_{i+2}, the attacker can inject two more false reports. More generally, an attacker controlling t compromised nodes can inject (or substitute, to be more precise) $t-1$ false data packets for every block of t authenticated reports.

The false data injection rate, however, is limited to the real event generation rate of the application. Moreover, this attack does not really defeat our design goal because it does not increase the traffic in the network. Although some false data packets are not filtered out on their way to the BS, the BS will detect them anyway. The energy expenditure of en-route nodes does not increase either because the number of reports, false or authentic, does not increase. Because of these reasons, we say this scheme provides a certain degree of immediate en-route filtering capability as in Scheme II.

6. FURTHER DISCUSSIONS

Table I compares both the security and the performance of the three schemes we presented above. We can observe that all these schemes have the same communication overhead. In practice, Scheme II or Scheme III should be selected because of their strong filtering capability. Compared to Scheme II, Scheme III is preferred because of its smaller computational overhead, although it requires additional memory space for buffering t reports. The additional memory space is usually not a big concern for a reasonable t. For example, if $t = 6$ and the size of a report is 29 bytes (as in TinyOS [Hill et al. 2000]), only 174 bytes are additionally needed.

Note that the choice of t should be based on both security and network node density. A large t makes it more difficult for an adversary to launch a false data injection attack, but it also results in more nodes being required to form a cluster. Moreover, we can separate the base station verification capability from the en-route filtering capability by using different values of the threshold t for them. For example, we can require a larger number of cluster nodes to endorse a report in order to provide stronger source authentication if the size of a cluster is large. Similarly, we may associate only a fraction of the cluster nodes with en-route nodes so that the overhead of en-route hop-by-hop authentication remains small. Finally, in practice, the en-route filtering functionality can be turned on or off as desired. For example, when no false data injection attack is detected, the base station can broadcast a command to turn off the en-route filtering, which will reduce both the computational and communication costs. When the base station receives false data reports, it can broadcast a command to turn on the en-route filtering.

7. PROTOTYPE IMPLEMENTATION

To study the practicality of our schemes for the current generation of sensors, we have implemented a prototype of one of our schemes on Berkeley Mica2 motes on

Table II. The required RAM space as a function of the system parameter t

t	1	2	3	4	5	6
RAM (bytes)	898	935	974	1015	1058	1103

top of the TinyOS platform [Hill et al. 2000]. Since all the three schemes involve three phases: association establishment, report endorsement, en-route filtering, and base station verification, we selected one of them, Scheme II, for implementation. The code for the scheme was written in nesC, a C-like language for developing applications on top of TinyOS.

We first derive an upper bound of t for the TinyOS platform with the default packet payload size of 29 bytes. Let the size of an IMAC and a PMAC be 8 bytes and 1 byte respectively, the size of an event 4 bytes, the size of an id 2 bytes, and the size of a cluster id 1 byte. Since the packet size of each one of the BS hello, CH acknowledgement, and data report messages is a linear function of t, we can easily calculate the largest t given the packet payload size. A CH acknowledgement message (similar to a BS hello message) contains a cluster head id, a cluster id, a MAC, and $t+1$ node ids. Therefore, for the message to fit into a single packet of 29 bytes, the maximum value of t is 8. On the other hand, a data report normally contains an event, an IMAC, $t+1$ PMACs, a cluster id, $t+1$ ids if the endorsing node set in the report is different with the previous one. In this case, the largest t is 4, which corresponds to a 28 byte payload. Therefore, the upper bound of t for TinyOS with 29-byte packets is $min(8, 4) = 4$. We note that for TinyOS the packet size may be increased, for example, from 29 bytes to 128 bytes. In the case of 128-byte packets, the upper bound of t is 38.

Our implementation includes code for all the four phases of our protocols as well as reliability mechanisms for endorsement messages. It also includes the Blundo scheme for pairwise key establishment[1]. On Mica2 motes [xbo 2005], the ROM needed for our code is 18.7 KB (out of the available 128 KB). The data memory used varies with the system parameter t. Clearly, if t is larger, a node has more association nodes, it has to store more pairwise keys, and more memory is needed. Table II shows the RAM used for data by our scheme as a function of t. Here the size of a key is 8 bytes and the sizes of an IMAC and a PMAC are 8 bytes and 1 byte respectively. We can see that data storage requirements of our scheme do not exceed the available RAM (4 KB) on Mica2 motes.

8. RELATED WORK

Przydatek, Song, and Perrig proposed SIA [Przydatek et al. 2003], a secure information aggregation scheme for sensor networks. SIA addresses the issue of false data injection using statistical techniques and interactive proofs, ensuring that the aggregated 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

[1]The code is available at http://www.cse.psu.edu/∼szhu/research/tinymesh.zip

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.

Hu and Evans [Hu and Evans 2003] propose a secure hop-by-hop data aggregation scheme that works if one node is compromised (i.e., $t = 1$). Ye et al [Ye et al. 2004] 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 schemes, when $t = 1$, a false data packet will be dropped immediately. Moreover, the packet overhead of our scheme is also smaller.

Perrig et al [Perrig et al. 2001] presented μTESLA for base station broadcast authentication, based on one-way key chains [Lamport 1981] and delayed key disclosure. Zhu et al [Zhu et al. 2003] 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. Indeed, the scheme in [Zhu et al. 2003] can be considered as a special case of our scheme where $t = 0$.

Recently, many key management schemes [Anderson et al. 2004; Chan et al. 2003; Chan and Perrig 2005; Du et al. 2003; Eschenauer and Gligor 2002; Liu and Ning 2003b; Zhu et al. 2003; Zhu et al. 2003] for sensor network security have been proposed . The polynomial-based pairwise key establishment scheme [Blundo et al. 1993] has been recently extended [Liu and Ning 2003b] to enable a sensor network to sustain more node compromises under the same memory constraints. Our schemes can also use other schemes if necessary although we demonstrated our scheme using the Blundo scheme.

Deng et al [Deng et al. 2003] discuss several security mechanisms for supporting in-network processing in hierarchical sensor networks. They also propose a multiple-base station and multiple-path strategy to increase intrusion tolerance, and an anti-traffic analysis strategy to disguise the location of a base station [Deng et al. 2004]. Karlof and Wagner [Karlof and Wagner 2003] describe several security attacks on routing protocols for sensor networks. Wood and Stankovic [Wood and Stankovic 2002] identify a number of DOS attacks in sensor networks. Different with most of these work, this work focuses on addressing a specific attack, the false data injection attack.

9. CONCLUSION AND FUTURE WORK

We presented several simple but effective authentication schemes to prevent false data injection attacks in sensor networks. The schemes guarantee 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 schemes enable an en-route node to detect and drop injected false data reports as early as possible, thus saving its energy that will otherwise be wasted for forwarding these false data reports. Our performance analysis shows the schemes are efficient with respect to the security it provides and allows a tradeoff between security and performance. Finally, we have demonstrated the feasibility of employing our schemes on resource-constrained sensor nodes by implementing one of the schemes on the TinyOS-based Mica2 motes.

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 [Karlof and Wagner 2003]. Another topic that we plan to address is how our scheme can be adapted for handling more complex data reports. In the presented schemes, $t+1$ nodes have to agree on an event to generate a report. In practice, a node's decision on an event may not be a boolean value because of the limitation of its detection ability. A node may agree on an event with certain level of confidence, i.e., with some probability. In this case, both the report endorsement phase and the en-route filtering phase will need to be modified. Also, our schemes do not work for hop-by-hop data aggregation applications. To address this problem, we will leverage our knowledge in designing SDAP, a secure hop-by-hop data aggregation protocol [Yi et al. 2006].

REFERENCES

2005. Crossbow technology inc.

Anderson, R., Chan, H., and Perrig, A. 2004. Key infection: Smart trust for smart dust. In *Proceedings of IEEE International Conference on Network Protocols (ICNP'04)*.

Bellare, M., Guerin, R., and Rogaway, P. 1995. Xor macs: New methods for message authentication using finite pseudorandom functions. In *Proceedings of CRYPTo'95*.

Blom, R. 1985. An optimal class of symmetric key generation systems. In *Advances in Cryptology, Proceedings of EUROCRYPT'84*. LNCS 209. 335–338.

Blundo, C., Santis, A. D., Herzberg, A., Kutten, S., Vaccaro, U., and Yung, M. 1993. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology, Proceedings of CRYPTO'92*. LNCS 740. 471–486.

Chan, H. and Perrig, A. 2005. Pike: Peer intermediaries for key establishment in sensor networks. In *Proceedings of Infocom'05*.

Chan, H., Perrig, A., and Song, D. 2003. Random key predistribution schemes for sensor networks. In *Proceedings of IEEE Security and Privacy Symposim'03*.

Deng, J., Han, R., and Mishra, S. 2003. Security support for in-network processing in wireless sensor networks. In *Proceedings of First ACM Workshop on the Security of Ad Hoc and Sensor Networks (SASN'03)*.

Deng, J., Han, R., and Mishra, S. 2004. Intrusion tolerance strategies in wireless sensor networks. In *Proceedings of IEEE 2004 International Conference on Dependable Systems and Networks (DSN'04)*.

Du, W., Deng, J., Han, Y., and Varshney, P. 2003. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*. 42–51.

Eschenauer, L. and Gligor, V. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of ACM CCS'02*.

Goldreich, O., Goldwasser, S., and Micali, S. 1986. How to construct random functions. *Journal of the ACM 33(4)*, 210–217.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. E., and Pister, K. S. J. 2000. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*. 93–104.

Hu, L. and Evans, D. 2003. Secure aggregation for wireless networks. In *Proceedings of Workshop on Security and Assurance in Ad hoc Networks*.

Karlof, C. and Wagner, D. 2003. Secure routing in sensor networks: Attacks and countermeasures. In *Proceedings of First IEEE Workshop on Sensor Network Protocols and Applications*.

Karp, B. and Kung, H. 2000. Gpsr: A geographic hash table for data-centric storage. In *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*.

Lamport, L. 1981. Password authentication with insecure communication communication. *Communications of the ACM 24(11)*, 770–772.

Liu, D. and Ning, P. 2003a. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03)*. 263–276.

Liu, D. and Ning, P. 2003b. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*. 52–61.

Liu, D., Ning, P., and Li, R. 2005. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security*.

Perrig, A., Szewczyk, R., Wen, V., Culler, D. E., and Tygar, J. D. 2001. Spins: security protocols for sensor netowrks. In *Proceedings of ACM Mobile Computing and Networking (Mobicom'01)*. 189–199.

Przydatek, B., Song, D., and Perrig, A. 2003. SIA: Secure information aggregation in sensor networks. In *Proceedings of ACM SenSys 2003*.

Rivest, R. 1994. The rc5 encryption algorithm. In *Proceedings of the 1st International Workshop on Fast Software Encryption*. 86–96.

Wood, A. and Stankovic, J. 2002. Denial of service in sensor networks. *IEEE Computer*, 54–62.

Ye, F., Luo, H., Lu, S., and Zhang, L. 2004. Statistical en-route detection and filtering of injected false data in sensor networks. In *Proceedings of IEEE Infocom'04*.

Yi, Y., Wang, X., Zhu, S., and Cao, G. 2006. Sdap: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the ACM Mobihoc*.

Zhu, S., Setia, S., and Jajodia, S. 2003. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*. 62–72.

Zhu, S., Xu, S., Setia, S., and Jajodia, S. 2003. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP'03)*.

Appendix A: The Blundo Scheme

The Blundo scheme was originally proposed by Blundo et al. [Blundo et al. 1993] 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 [Blundo et al. 1993]; 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.