A Fault Localized Scheme for False Report Filtering in Sensor Networks

Li Zhou and Chinya V. Ravishankar
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92521, USA
{lzhou,ravi}@cs.ucr.edu

Abstract

Sensor networks frequently deploy many tiny and inexpensive devices over large regions to detect events of interest. It can be easy to compromise sensors, enabling attackers to use the keys and other information stored at the sensors to inject false reports, forging fake events. Existing approaches do not localize the impact of such node compromises, so that compromises in one sensing region may compromise other parts of the system. In this paper, we propose two fault localized schemes for false report filtering. In our basic scheme, sensors signal events using one-way hash chains, which allows en-route nodes to verify the authenticity of received reports based on commitments of detecting sensors, but prevents them from forging events. We extend this basic scheme to a collaborative filtering scheme using commitment predistribution, making it more adaptable for mobile sensor networks and high-density sensor networks. Our scheme can also provide localized protection for areas that require special protection. Our security analysis shows that our schemes can offer stronger security protection than existing schemes, and are efficient.

Keywords:

wireless sensor networks, security services, false report attacks, fault localization, mobility.

1 Introduction

Sensor networks commonly deploy large numbers of sensors over a large field to sense events or obtain readings of various parameters. Such networks may be deployed, for example, to detect forest fires, unusual traffic patterns, environmental changes, or troop movements in battle. Typically, a sensor detecting an event sends a report to a special node called a *sink*, which collects and processes such reports. These reports must be delivered securely to the sink in cases where node compromises are likely, or when the environment is hostile. Unfortunately, it is hard to secure sensors physically, since they are prone to capture, and strong cryptographic methods can require more resources than are typically available. Upon compromising a node, the adversary gains access to all data and cryptographic keys stored therein, and may have enough key information to impersonate the compromised node and mount a variety of attacks.

We present a mechanism to address false report generation attacks [13], which are mounted by an adversary who compromises a node and generates false reports appearing to originate from it. False reports waste scarce resources such as energy and bandwidth, but more importantly, they may trick the sink into making wrong decisions, with serious high-level consequences. For example, a fake report of an emergency might send first-responders to the wrong location, depriving legitimate emergencies of urgently-needed resources. Consequently, false reports should be filtered as early as possible as they travel to the sink.

Such a scheme must localize faults, and be efficient and scalable. Fault localization requires that the compromise of some sensors not render other parts of sensor network ineffective. As we will see, this issue has not been addressed by current approaches. Efficiency requires the scheme to consume few resources and to filter false reports as early as possible. Scalability is important because sensor deployments can be very large.

1.1 Our Work

In this paper we present two efficient, scalable, and fault-localized schemes for false report filtering. In our basic scheme, each detecting sensor signals events using a one-way hash chain, so that en-route nodes can verify the authenticity of reports using key commitments by detecting sensors. Unlike earlier work [13, 16], our scheme differentiates between the roles of detecting nodes and en-route nodes, and limits the impact of a node compromise to its locale. As a consequence of fault localization, this scheme also enables localized protection when some important areas require special protection. We develop a collaborative false report filtering scheme using commitment predistribution, which is suitable for mobile sensor networks and adaptable to high-density sensor networks.

Our security analysis shows that more than 98% false reports are dropped within 2 hops when false reports are generated by a compromised cluster head. In the worst case, where false reports are collusivly forged by a threshold number of detecting nodes, more than 90% false reports can be filtered within 8 hops. Further, although our scheme introduces extra fields in report packet, it results in very considerable energy savings by filtering most false reports very quickly.

The rest of this paper is organized as follows. We describe the related work in Section 2. Section 3 motivates our work. We present and analyze our basic fault-localized scheme for false report filtering in Section 4, and present and analyze an enhanced scheme for supporting mobile sensor networks in Section 5. And we evaluate their performance in Section 6. Finally, we make a conclusion in Section 7.

2 Related Work

Recent approaches [13, 16, 12] to false report filtering have assumed that the sensor are densely deployed, and that there are no more than some number t of system-wide node compromises. Forwarding nodes simply drop any report with t or fewer endorsements before it reaches the sink.

In [13], endorsements take the form of Message Authentication Codes (MACs) generated using keys assigned randomly to nodes from a global key pool divided into disjoint partitions. Keys from more than t key partitions are not assumed to be compromised, so that a report must include MACs using keys from more than t different key partitions to be considered valid. Randomly assigning a sufficient number of keys from the global pool to each node leads to a high probability that en-route nodes share keys with detecting nodes, and can verify endorsements. This scheme is effective in filtering false reports when the number of key partitions that compromised keys come from is no more than t. However, since each reporting region must include keys from $t+1$ or more partitions, physical compromise of any reporting region automatically yields keys from at least $t+1$ partitions. Since the keys are drawn from the same global pool, this creates a serious problem, since the number of key partitions that compromised keys come from now exceeds t, and an attacker can fabricate false reports appearing to originate from arbitrary reporting regions, rendering the entire network ineffective. This lack of fault localization causes serious security problems.

Zhu et al [16] present an interleaved hop-by-hop authentication scheme, in which each node is associated with two other nodes on the path from the cluster head to the sink, called the lower association node and upper association node, respectively. An en-route node will forward received reports if successfully verified by its lower association node. The security of this scheme depends mainly on the correct creation of associations, which cannot be guaranteed. In the worst case, t compromised nodes can force a false report to bypass verification by as many as $O(t^2)$ uncompromised nodes. Besides, as pointed out in [12], the work in [13, 16] uses symmetric keys allowing compromised nodes to abuse compromised symmetric keys to generate false reports.

Yang et al. [12] propose a commutative cipher based scheme, in which en-route nodes simply check whether a report originates at one of the cluster heads specified by the sink, letting the sink eventually verify the correctness of the cluster heads. When cluster heads are compromised, en-route nodes cannot detect false reports, and will forward them to the sink. This is inefficient and wastes considerable energy. Further, when this scheme is extended to deal with general queries that do not specify location, it requires the sink to flood a probe message before an event occurs. This is unreasonable since events are unpredictable.

3 Motivation and Assumptions

We will now discuss two ideas that are central to our work: *fault localization* and *localized protection*.

3.1 Fault Localization

Sensors are prone to physical capture or compromise, and keys and other information stored at the nodes become available to adversaries upon such compromise. Without fault localization, the compromise of some sensors can compromise others.

In SEF [13], a valid report must include $t+1$ MACs, computed with keys drawn from different key partitions of a global pool. Since reports are certified by other nodes in the physical neighborhood of a reporting node, each reporting neighborhood must contain nodes whose keys come from more than t such partitions. SEF remains safe only if attackers can not compromise keys in more than t key partitions. However, in hostile environments, this assumption is unrealistic, and it may be possible for an adversary to compromise an entire reporting region. Since the nodes in each such region must have keys from more than t partitions, the adversary will immediately acquire keys from more than t partitions. These keys may be used to compute more than

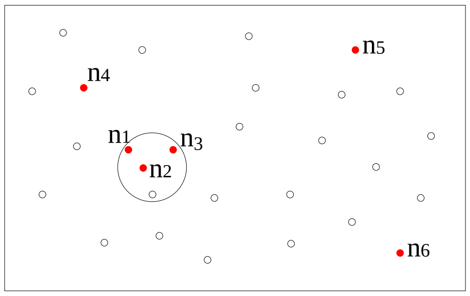


Figure 1. Fault Localization and Localized Protection

t valid MACs, enabling the adversary to fake reports *anywhere* in the system. Even worse, these false reports will not be filtered either by en-route nodes or by the sink, since they include $t + 1$ valid MACs. Therefore, node compromise in one small detecting area may lead to compromise of the whole network. In Figure 1, let $t = 2$ and let nodes n_1, n_2 and n_3, holding keys from different partitions, be compromised. Now, the attacker can not only fake a false report notifying the occurrence of an event in n_1's detecting area, but also fake reports from anywhere.

In contrast, our scheme is safe as long as no more than t sensors are compromised in any one detecting area. Further, the compromise of more than t sensors in any detecting area only allows the attacker to fabricate events in that detecting area. The security of other detecting areas will remain unaffected.

In LEAP [16], reports include MACs computed with keys shared either between detecting sensors and en-route sensors or between a pair of en-route sensors, enabling compromised en-route sensors to forge false reports and collusivly forward them through up to $O(t^2)$ uncompromised en-route sensors. Although these false reports will be eventually detected by the sink, they consume scarce energy at en-route sensors. In contrast, we distinguish between the functions of detecting nodes and en-route nodes, so such collaborations are impossible.

3.2 Localized Protection

This feature is a consequence of fault localization. In wireless sensor network applications such as military deployments, there may be important regions that need special protection. A scheme with localized protection allows protection of such selected areas, enabling the best use of resources of entire network.

In Figure 1, for example, let node n_1's detecting area require special protection, and be specially hardened against compromises. In a scheme such as SEF [13], nodes are assigned keys from randomly selected partitions. Suppose nodes in each of the pairs (n_1, n_4), (n_2, n_5), and (n_3, n_6) are assigned keys from the same key partition. Although the special protection may make it hard to compromise nodes n_1, n_2 and n_3, it suffices instead to compromise nodes n_4, n_5 and n_6 outside this protected area. An adversary can now fabricate reports from n_1's detecting area without being detected. Thus to ensure the safety of n_1's detecting area, protection should be performed uniformly across the entire network, requiring an unacceptable amount of resources in current schemes. For a large scale sensor network, offering localized protection will be a great challenge.

The schemes in [13, 16] are not able to achieve fault localization and localized protection, because they do not differentiate between the roles of detecting nodes and en-route nodes. Detecting nodes are used to generate valid reports, while en-route nodes are used to forward verified reports to the sink, and to filter false reports as early as possible. While both [13] and [16] enable en-route nodes to use shared keys with detecting nodes to both verify and generate reports, they also make it possible for compromised en-routing nodes abuse their shared keys to fabricate reports. In contrast, our schemes enable en-route nodes only to verify received reports based on commitments from detecting sensors, but not to generate reports.

3.3 Replay Attacks

The schemes in [13, 16] are all vulnerable to replay attack, in which compromised nodes record and replay legitimate reports. It is harder for en-route nodes or even the sink to detect such report replays than to detect false reports. In previous schemes [13, 16] en-route nodes and the sink can use keys shared with the detecting nodes to verify the correctness of received reports, but not their freshness. Even when the sink eventually recognizes false reports, they contain insufficient information to locate adversaries.

3.4 Our Assumptions

We assume that the sink has sufficient resources to protect itself against compromise. Sensors, on the other hand, are small battery-equipped devices with limited computation and storage capability, and subject to capture or compromise. Adversaries can eavesdrop, or inject, modify or replay messages transmitted in the network. They may also collude in performing such compromises. Further, as in [16, 12], we assume that the transmission range of the sink is the same as that of the sensors. Consequently, messages from the sink may travel to sensors over several hops.

We assume that no more than t nodes are compromised within any cluster. While compromised nodes may mount a variety of attacks, such as dropping valid reports or not

Notation	Description
N	the size of the sensor network
N_G	the size of a cluster
N_c	the number of compromised sensors
η	the number of sensors in a sensor's transmission range
n_i	the id of a sensor
t	an event is recognized as legitimate when at least $t+1$ sensor detects it
K_i	a secret key shared between sensor n_i and the sink
K_{hi}	the one-hop pairwise key shared between n_i and its cluster head
c_i^p	the pth hash value on the one-way hash chain of n_i

Table 1. Our Notation

reporting the events that they detect, addressing such attacks requires active monitoring of transmissions, and is beyond the scope of this paper. Our focus, as in [13, 16, 12] is on filtering false reports generated by compromised nodes before they reach the sink.

Finally, we assume that paths between the sink and the sensors are symmetric, so that the same nodes are traversed by messages going in either direction, albeit in opposite orders.

Table 1 lists the notations which appear in the rest of this paper.

4 Our Scheme

A central idea in our scheme is to differentiate between the roles of detecting nodes and en-route nodes. Detecting nodes detect and report events of interest, while en-route nodes verify received reports and forward verified reports to the sink. En-route nodes are not given access to the keys used to generate reports, so that they are prevented from abusing keys to generate false reports. This feature differentiates our work from that in [13, 16].

In our scheme, each detecting sensor maintains a one-way hash chain, and signals an event E of interest by disclosing the next available hash value on its chain. It also computes a MAC on E using a pairwise secret key shared with the sink, so that the sink can verify the authenticity and integrity of the report. The network is divided into clusters, and a cluster head is selected for each detecting area. Since no more than t sensors may be compromised in any detecting area, an event must be endorsed by $t+1$ sensors. When the cluster head collects $t+1$ hash values, representing $t+1$ agreements on an event occurrence, it generates and forwards to the sink a report which includes these hash values and their corresponding MACs. If an en-route node on the path from the cluster head to the sink holds the hash chain commitment of any node endorsing this event, it verifies the hash value reported by the endorser.

We note that en-route nodes can not fake reports using the hash-chain commitments for detecting nodes, since reports are recognized as legitimate only when they contain fresh values from the hash chain. Since fresh hash chain values can not be generated from the public commitments, fake reports generated by en-route nodes can be easily detected and filtered by uncompromised en-route node.

Our scheme includes four phases: sensor initialization and deployment, report generation, en-route verification, and sink verification. We describe the details of each phase in the following sections.

4.1 Sensor Initialization and Deployment

Before a sensor node n_i is deployed, it is initialized with a pairwise secret key K_i shared with the sink and used to authenticate the messages sent from n_i to the sink, as well as a one-way hash chain $C_i = c_i^0, c_i^1, \cdots, c_i^m$. A one-way hash chain [5] is generated using a one-way hash function $\mathcal{F}$, so that

$$c_i^j = \mathcal{F}(c_i^{j+1}), \quad j = 0, 1, \ldots, m-1$$

To save memory, each sensor may choose to store parts of this chain. For example, it may only store every kth hash values, deriving the rest using the one-way hash function. This is a tradeoff between the memory and computation.

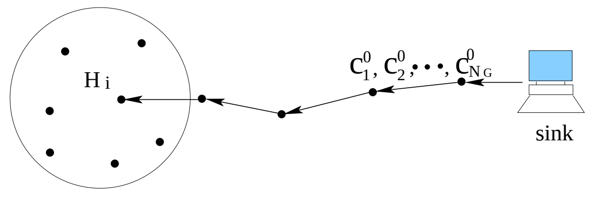
The hash values in this chain are used in reverse order, starting with c_i^1. The value c_i^0 serves as the commitment for the hash chain, and is made public, so that other sensors are able to verify the correctness of received hash values. Given commitment c_i^0, a node can verify any hash value c_i^k by checking whether $c_i^0 = \mathcal{F}^k(c_i^k)$.

Initially, the network is partitioned into clusters, called detecting areas. Each cluster contains at least $t+1$ detecting sensors, the size of the clusters being determined by the application requirement and the sensor density. A cluster head is elected, and periodically rotated to enhance robustness. Cluster head election algorithms are out of the scope of this paper, and readers are referred to other work [10, 14] for schemes to accomplish this task. After a sensor is deployed, it establishes pairwise keys with its one-hop neighbors using schemes proposed in [2, 6, 7, 15, 1].

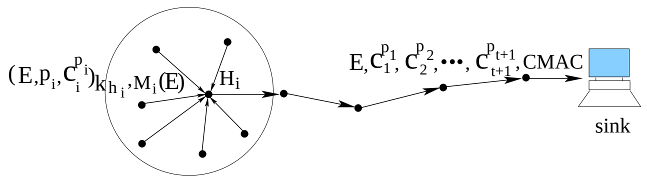
To enable en-route nodes to verify reports, the sink transmits the commitments for sensors for cluster i to the corresponding cluster head H_i. This message is authenticated using μTESLA [9]. If the size of a cluster is N_G, each en-route node randomly stores k out of N_G commitments based on its storage capability. The more it stores, the better it is able to verify reports. If an en-route node is on multiple paths to the sink, it must store commitments for each cluster.

$$\forall i : sink \rightarrow H_i : c_1^0, c_2^0, \cdots, c_{N_G}^0$$

After this transmission, a path is generated between each cluster head and the sink, to be used for secure report transmission. When these paths change due to node failure, we can use the local repair scheme or the sink initiated repair scheme proposed in [16] to repair these paths. A path is repaired locally by replacing the failed nodes with good ones. Since these new nodes do not have the commitments needed to verify reports, they just forward all received reports. When the path is repaired by the sink, the commitments will be piggybacked in every beaconing message [4] sent by the sink.



(a) Initialization


$(E, p_i, C_i^{p_i})_{K_{h_i}}, M_i(E)$
H_i
$E, C_1^{p_1}, C_2^{p_2}, \cdots, C_{t+1}^{p_{t+1}}$, CMAC
sink

(b) Report Generation

Figure 2. Initialization and report generation

Determining k: The number of commitments each en-route node stores for a specified cluster is restricted by the memory capacity of sensors, and dictates the ability of en-route nodes to verify reports. Suppose the size of a cluster is N_G, and a valid report is to include $t+1$ hash values from distinct detecting sensors. To enable an en-route node to verify reports, it must hold a commitment for at least one of the $t+1$ hash values included in the report, so the value of k should satisfy

$$N_G - k \leq t,$$

so that $k \geq N_G - t$. For example, if the size of a cluster is $N_G = 7$, and a valid report is to include 5 endorsements on an event, the value of k must be at least 3.

As we show in Section 5, en-route nodes that are on multiple paths might not have enough memory to store the required commitments in a large high-density sensor network. We present a collaborative scheme to solve this problem in Section 5.

4.2 Report Generation

When a sensor n_i detects an event of interest, it sends a report to its cluster head H_i.

$$n_i \rightarrow H_i : [E, p_i, c_i^{p_i}]_{K_{hi}}, MAC(K_i, E)$$

E specifies the type of the event and other relevant information. $c_i^{p_i}$ is the p_ith hash value, which is the next available hash value, on detecting sensor n_i's chain. This value denotes n_i's endorsement of the occurrence of E. En-route nodes may verify n_i's endorsement by checking the correctness of this hash value $c_i^{p_i}$. To enable the sink to verify the authenticity and integrity of its report, n_i also computes a MAC on E using the secret key K_i it shares with the sink. The authenticity and integrity of E, p_i and $c_i^{p_i}$ are ensured by K_{hi}, a pairwise key shared between n_i and H_i. We do not use μTESLA, since it consumes up one distinct hash value at each time interval, whether or not an event occurs, and approach that wastes precious hash values. In our scheme, each detecting sensor discloses a hash value only when it detects an event.

An event is recognized as legitimate when at least $t+1$ sensors detect it. When the cluster head collects at least $t+1$ agreements from detecting sensors, it sends to the sink a report that includes the $t+1$ hash values sent by distinct detecting sensors, and the corresponding $t+1$ MACs. The hash values are used for en-route verification, while the MACs are used for final verification at the sink. The IDs of the $t+1$ detecting nodes and the indices of hash values are also included in the report, so that en-route nodes and the sink can verify the received reports. Rather than including a series of individual message authentication codes $M_i(E) = MAC(K_i, E)$, a cumulative message authentication code (CMAC) $M^{\oplus}(E)$ is obtained by computing an exclusive-or on the $M_i(E)$ as follows.

$$M^{\oplus}(E) = M_1(E) \oplus M_2(E) \oplus \cdots \oplus M_{t+1}(E)$$

$$\begin{aligned} H_i \rightarrow sink : {} & n_1, n_2, \cdots, n_{t+1}, p_1, p_2, \cdots, p_{t+1}, \\ & c_1^{p_1}, c_2^{p_2}, \cdots, c_{t+1}^{p_{t+1}}, M^{\oplus}(E) \end{aligned}$$

4.3 En-route Verification

When an en-route node receives a report, it first checks the number of hash values included in the report. If the number of included hash values is less than $t+1$, it drops the packet, since it does not include enough agreements. Otherwise, if an en-route node u stores commitments for x of the detecting nodes that endorse this report, it can verify x hash values included in the report. Specifically, suppose u

stores detecting sensor n_i's commitment. When u receives a report which includes p_i and $c_i^{p_i}$, it verifies $c_i^{p_i}$ in two steps. First, u checks whether p_i, the index of $c_i^{p_i}$, is larger than the index of the value it previously received, say, p_i', confirming that $c_i^{p_i}$ is a new hash value. Second, for a fresh value $c_i^{p_i}$, u checks whether $c_i^{p_i'} = \mathcal{F}^{p_i - p_i'}(c_i^{p_i})$. If these two checks succeed, the en-route node relays the report to the next node on the path to the sink.

In our scheme, we do not assume that cluster heads are secure, or that they are hardened, or have special attributes. A compromised cluster head may drop or manipulate true reports, or inject false reports. In this paper, we focus on false report generation attacks. Denial of service attack mounted by compromised cluster heads by dropping reports require special monitoring, and are not addressed here.

To inject a false report, the compromised cluster head must fabricate t hash values and a cumulative MAC. An uncompromised en-route node that has the commitment of any fabricated hash value will filter this false report immediately. Further, since each en-route node maintains the indices of hash values, replaying old reports will also be detected by an uncompromised en-route node.

4.4 Verification at Sink

When the sink receives a report, it checks the number of hash values included, and their correctness. It also computes $t + 1$ MACs using keys shared with each detecting node, and XORs them. If the cumulative MAC matches the one included in the report, the sink accepts the validity of the report. Otherwise, it has identified a set of misbehaving nodes, and takes corrective action against them.

4.5 Security Analysis

We now show that our scheme performs well in terms of its ability to address false report generation, and is efficient.

4.5.1 Outsider Attacks

False reports generated by outside attackers can be filtered by any uncompromised nodes immediately. Replaying old reports can also be detected, because the indices of hash values along the hash chain allow the forwarding nodes to verify the freshness of reports.

4.5.2 Insider Attacks

Unlike previous schemes [13, 16], en-route nodes in our scheme are only able to verify reports using the commitments of detecting nodes, but not generate fake reports, since they cannot fabricate valid hash values belonging to the hash chains of other detecting nodes. Compromised

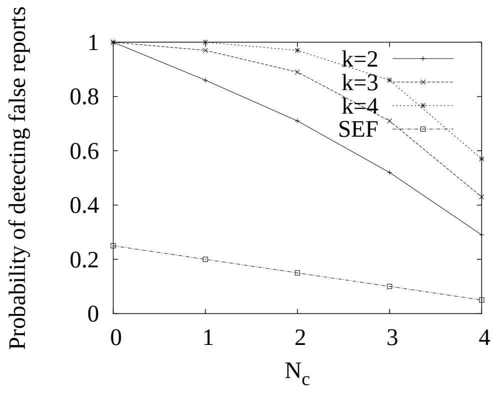


Figure 3. The probability of a false report being filtered within one hop

en-route nodes must forge $t + 1$ hash values of detecting nodes to generate a false report. These will be detected and dropped by an uncompromised en-route node in our scheme.

Let $\delta(h)$ be the probability that a false report will be filtered in h hops. Let there be N_G detecting nodes in a cluster, and let N_c of these, including the cluster head, be compromised. To forge a report, the cluster head must forge $t + 1 - N_c$ hash values. If each en-route node holds k commitments for that cluster, the detection probability is the probability that an en-route node happens to have commitment for at least one those $t+1-N_c$ forged hash values. Clearly,

$$\delta(1) = 1 - \frac{\binom{N_G - t - 1 + N_c}{k}}{\binom{N_G}{k}}.$$

Figure 3 shows the probability of a false report being filtered within one hop, when the size of the cluster is 7 and an event is recognized when $t+1 = 5$ detecting sensors endorse its occurrence. If each en-route node stores two commitments for that cluster, a false report from a compromised cluster head ($N_c = 1$) will be filtered within one hop with a probability of 86%. When four detecting nodes including the cluster head have been compromised, false reports will be filtered within one hop with the probability of 29%.

As Figure 3 shows, the probability of a false report being filtered within one hop increases with the number of commitments that each en-route node stores. When each en-route node stores seven commitments for that cluster, it suffices to have a single uncompromised en-route node to filter all false reports.

Figure 3 also shows the probability of false reports being filtered within one hop in SEF. However, we should note that SEF can not be directly compared with our scheme, since the meanings of N_c in two schemes are different. N_c in our scheme denotes the number of compromised sensors,

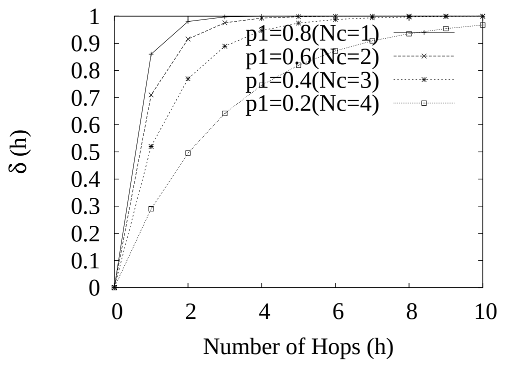


Figure 4. The portion of dropped false reports as a function of the number of traveled hops.

while in SEF, N_c denotes the number of key partitions that compromised keys come from.

The expected fraction of false reports detected and dropped within h hops is

$$\delta(h) = 1 - [1 - \delta(1)]^h = 1 - \left[\frac{\binom{N_G - t - 1 + N_c}{k}}{\binom{N_G}{k}} \right]^h$$

Figure 4 illustrates that the fraction of dropped false report increases rapidly as the hop count grows. We assume here that an event is legitimate when at least five nodes have detected it. Figure 4 shows that more than 98% false reports are dropped within two hops when only the cluster head is compromised. In the worst case, where a threshold (4 in this example) number of detecting nodes have been compromised, 90% of false reports are dropped within 8 hops.

5 High-Density Deployments & Mobility

In our basic scheme, en-route nodes are able to filter most of false reports in a few hops, using commitments for detecting sensors. Now, we consider the problem of false report filtering in two scenarios: high-density sensor networks and mobile sensor networks.

Scenario I: High-Density Deployments: A common approach to extending the lifetime of sensors is to divide each cluster into y groups, and activate each group in turn [11, 3]. This ensures that each sensor is active for $1/y$ of the total time. To ensure that event will be reported reliably, each group must have at least $t+1$ sensors. Thus a high-density sensor network is required to satisfy this application requirement.

Our basic scheme is not suitable for high-density sensor networks, since the storage requirement becomes excessive. For example, consider a $5,000$-sensor network, in which each cluster has 100 sensors divided into five 20-sensor groups. Suppose a legitimate event must be detected by at least $t+1 = 11$ sensors. As shown in Section 4.1, each en-route node must store at least $N_G - t$ commitments to enable itself to verify the received reports. Under the above configuration, each en-route node needs to store 90 commitments per cluster. If an en-route node is on multiple paths to the sink, say 6 paths, it needs around $4.3KB$ to store commitments. To put this in perspective, the MICA2 Mote [8] only offers $4KB$ SRAM.

Scenario II: Mobile Networks: Consider a mobile scenario, in which the sink may move through the entire target region while the sensors are static after deployment. An example of this situation is sensors deployed in a battlefield, where the sink is deployed in a vehicle that moves across the battlefield.

When the sink moves to a new location, the paths from each cluster to the sink are all changed accordingly. The sink must send commitments and latest indices of the hash values to each cluster head again, enabling en-route nodes to update the commitment-index pairs they need store.

$$\forall i : sink \rightarrow H_i : n_1, n_2, \cdots, n_{N_G}, c_1^0, c_2^0, \cdots, c_{N_G}^0, \\ p_1, p_2, \cdots, p_{N_G}^0$$

Consider a $1,000$-sensor network, and let the node ID and the indices all be 10 bits, and the length of each commitment be 64 bits. Whenever the sink moves, it will send out around $8KB$ in commitments and $1.2KB$ in indices. If the sink moves frequently, sending these messages repeatedly will waste scarce bandwidth and energy. In this section, we present a collaborative false report filtering scheme, enabling the sink to only send the short-length indices, and thus saving lots of bandwidth and energy consumption.

The basic idea in this scheme is for every sensor to be randomly preloaded with m out of N commitments, where m is a system parameter and N is the number of nodes in the network. When the sink moves to a new location, it only sends the latest indices of hash values of sensors to their cluster head. Each en-route node stores the indices, and will collaborate with its one-hop neighboring sensors to verify the received reports.

This scheme works in four phases: sensor initialization and deployment, report generation, en-route verification, and sink verification. The difference between our two schemes is the method of distribution of commitments in the first phase, and en-route verification in the third phase. The other two phases are the same, so we only describe the first and third phases of this enhanced scheme.

5.1 Sensor Initialization and Deployment

As in our first scheme, each sensor is preloaded with a secret key shared with the sink, and a one-way hash chain. Unlike the first scheme, however, each sensor is preloaded with m commitments prior to its deployment. m is a system parameter determined by the memory capability of sensors, and directly affects the verification ability of each node. We give a detailed analysis in Section 5.3.

To enable en-route nodes to verify reports, the sink sends the indices of the latest hash values on the corresponding hash chains to each cluster head.

$$\forall i : sink \rightarrow H_i : n_1, n_2, \cdots, n_{N_G},\ p_1, p_2, \cdots, p_{N_G}$$

When an en-route node u receives this message, it first checks whether it holds any commitments for nodes specified in the message. If it does, it records the corresponding indices. Otherwise, it sends a request to its neighboring sensors, asking whether they have the commitments for the specified nodes. The node u will store any indices that their corresponding commitments are held by u's neighboring nodes.

Whenever the sink moves to a new location, the sink sends the latest indices of hash values of sensors to their cluster heads, enabling en-route nodes to update their indices.

5.2 En-route Verification

As in our first scheme, each en-route node will first check the number of hash values included in a received report, verifying whether this report is endorsed by at least $t+1$ detecting sensors. Unlike the first scheme, however, each en-route node will now first check whether it has the commitment-index pair of any one of the $t+1$ hash values. If it does, it verifies the corresponding hash value(s) as in the first scheme. If it holds indices but not commitments, for any of the $t+1$ hash values, the en-route node u contacts its one-hop neighboring sensors, asking for the commitment(s) of those hash values. u verifies the corresponding hash values using these responses and the indices it stored in the first phase.

5.3 Security Analysis

The security analysis for outsider attacks of the enhanced scheme is identical to that of the basic scheme (Section 4.5.1). In this section, we analyze the security of the enhanced scheme in terms of its ability to filter the false reports generated by compromised detecting sensors.

In our basic scheme, each en-route node is guaranteed to have the commitment-index pair corresponding to at least one of the hash values included in the received report. However, since we now randomly preload m commitments, we can not guarantee that an en-route node and its neighbors will have the commitment-index pair for any hash value. Let the size of the network be N, the expected number of sensors within a sensor's transmission range be η, and a valid report include $t+1$ agreements. Let α be the probability that a node and its 1-hop neighbors hold at least one of the commitments for the $t+1$ hash values in a message. Clearly,

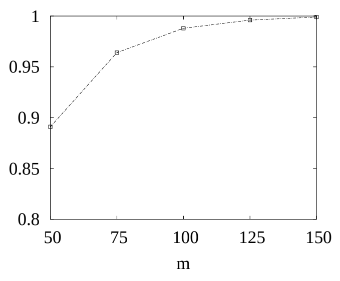


Figure 5. The probability of an en-route node and its neighboring nodes having at least one commitment of the $t+1$ hash values included in a report ($N = 5,000$, $t = 10$, $\eta = 20$)

$$\alpha = 1 - \left[\frac{\binom{N-t-1}{m}}{\binom{N}{m}} \right]^{\eta}$$

Figure 5 shows that an en-route node and its neighboring nodes will hold at least one commitment of the $t+1$ hash values with a very high probability. For example, when each sensor is preloaded with 100 commitments, this probability is 99%, suggesting that each en-route node and its neighboring nodes can nearly always collaboratively verify the received reports.

Now we consider the probability of a false report being filtered in one hop. Let $A(k)$ be the event that a false report is filtered in k hops, and $B(i)$ be the event that a node and its 1-hop neighbors hold i commitments for nodes in a specified cluster. Now,

$$\Pr[A(1)] = \sum_i \Pr[A(1) \mid B(i)] \cdot \Pr[B(i)].$$

Now, $\Pr[A(1) \mid B(i)] = 1 - \binom{N_G - t - 1 + N_c}{i} / \binom{N_G}{i}$, as in Section 4.5.2.

Let γ be the probability that any given commitment c is held by a node or its 1-hop neighbors. Now,

$$\gamma = 1 - \left[1 - \frac{\binom{N-1}{m-1}}{\binom{N}{m}} \right]^{\eta}.$$

Consequently,

$$\Pr[B(i)] = \binom{N_G}{i}\gamma^i(1-\gamma)^{N_G-i},$$

so that

$$\Pr[A(1)] = \sum_{1\le i\le N_G}\left[1 - \frac{\binom{N_G-t-1+N_c}{i}}{\binom{N_G}{i}}\right] \times \binom{N_G}{i}\gamma^i(1-\gamma)^{N_G-i}.$$

For example, consider a sensor network with $N = 1,000$ sensors, and is divided into 7-sensor clusters. There are $\eta = 10$ sensors in a sensor's transmission range. A valid report includes 5 endorsements. A false report generated by two compromised detecting sensors will be filtered in one hop with the probability of 95.7%.

6 Performance Evaluation

We evaluate the performance of both our schemes in terms of the energy savings from false report filtering, as well as storage requirements.

6.1 Energy Savings

In our scheme, sensors consume energy in two ways: (1) to transmit reports, and (2) to verify reports by computing hash values. Since the sink is assumed to have sufficient resources, we ignore the energy consumed at the sink. Besides, as shown in [9], computing hash values increases energy consumption only marginally. Thus, we ignore the energy overheads of computation, and concentrate only the communication overhead.

When an en-route node u first receives a report endorsed by sensor n_i in our enhanced scheme (Section 5.2), it may need to ask its neighbors for the commitment c_i^0 of n_i, if u stores the latest index of the hash chain of n_i. This is a one-time request. Once u gets and stores c_i^0, it can verify n_i's hash values appearing in all future reports. This one-time local communication cost is amortized over all report transmissions, and introduces marginal additional overhead. In other words, the energy consumed for communication during report transmission in our enhanced scheme is marginally higher than that of our basic scheme. Due to space limitations, we provide an analysis of energy savings for our basic scheme.

Our scheme requires each report to include extra $t + 1$ node IDs, $t+1$ hash values and their corresponding indices, and a cumulative MAC, so that it involves extra energy consumption for communication. However, as Section 4.5.2 illustrates, our scheme is able to filter most of false reports within a few hops, significantly reducing the energy wasted by false reports.

We use a model similar to that in [13] to quantify the energy consumption. Let L_r, L_n, L_k, L_s and L_{CM}, denote the lengths, respectively, of a regular report without our security mechanism, the node id, the hash value index, the hash value, and of the compressed MAC. The length of a report packet in our scheme is $L'_r = L_r + L_{CM} + (L_n + L_k + L_s)\cdot(t+1)$, and is normalized to original report length as $\frac{L'_r}{L_r} = 1 + \frac{L_{CM}}{L_r} + \frac{L_n+L_k+L_s}{L_r}\cdot(t+1)$. Let the number of hops a report travels be H, and the normalized legitimate traffic and fabricated traffic be 1 and β. Let the energy consumed for delivering all reports without our security mechanism be E_r, and the energy consumed with our security mechanism in place be E_s. Now,

$$E_r = H(1+\beta),$$

and

$$E_s = \left[1 + \frac{L_{CM}}{L_r} + \frac{L_n+L_k+L_s}{L_r}\cdot(t+1)\right] \times H\left[1 + \beta\cdot\frac{1-(1-P_1H)(1-P_1)^H}{P_1}\right].$$

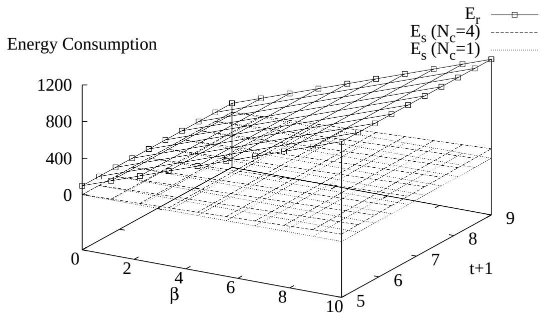


Figure 6. Energy Consumption

Figure 6 compares the energy consumption for report transmission with and without our security mechanism, when the original packet size is $L_r = 24$ bytes, node id is $L_n = 10$ bits, hash value index is $L_k = 10$ bits, the length of hash value is $L_s = 64$ bits, the length of compressed MAC is $L_{CM} = 64$ bits, and $H = 100$ hops.

Although our scheme introduces extra fields in report packets, Figure 6 shows that our scheme results in very considerable energy savings.

6.2 Storage Requirements

Our schemes mainly require each sensor to store its one-way hash chain and the commitments of hash values of some other detecting sensors to verify the received reports. As noted in Section 4.1, each sensor can store parts of its hash chain, deriving the rest of it using the one-way hash function $\mathcal{F}$. Suppose each sensor stores n_1 hash values on its chain.

In our basic scheme, each en-route node needs to store $N_G - t$ commitments to ensure that it is able to verify reports from the specified cluster to the sink. If an en-route node is on l paths to the sink, it must store $l * (N_G - t)$ commitments. For example, suppose the size of the clusters is $N_G = 7$, $n_1 = 100$, $l = 20$, $t = 4$, and the length of each commitment is $L_s = 64$ bits, the total storage requirement of this scheme is around $1.2KB$. As we showed in the beginning of Section 5, this scheme is not suitable for high-density sensor networks, since the storage requirement is nearly proportional to the size of the clusters, which is decided by sensor density.

In the enhanced scheme, as we showed in Section 5.3, an en-route node with 100 preloaded commitments is able to verify almost any report from the specified clusters to the sink. Thus the total storage requirement is around $100 * 8B + 100 * 8B \approx 1.5KB$. The MICA2 Mote [8] offers $4KB$ of SRAM. Therefore, the storage requirements of our schemes are very reasonable.

7 Conclusion

In this paper, we present two fault-localized schemes for false report attacks in wireless sensor networks. Our schemes differ from existing schemes since they differentiate between the roles of detecting nodes and en-routing nodes, and enable en-route nodes only to verify received reports based on commitments from detecting nodes, but not to generate reports. Therefore, the impact of a node compromise is limited to its locale. We generalize our basic scheme to a collaborative false report filtering scheme using commitment predistribution, making it more suitable for mobile sensor networks and adaptable for high-density sensor networks. Our security and performance analysis shows that our schemes are able to filter most of false reports within a few hops, significantly reducing the energy consumption due to the false reports.

Acknowledgement

This work is supported in part by grants from Tata Consultancy Services Inc., and the Fault-Tolerant Networks program of Defense Advanced Research Projects Agency, under contract F30602-01-2-0536.

References

[1] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM*, 2004.

[2] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM Conference on Computer and Communication Security(CCS)*, 2003.

[3] H. Gupta, S. Das, and Q. Gu. Connected sensor cover: self-organization of sensor networks for effi cient query execution. In *ACM International Symposium on Mobile Ad Hoc Networking & Computing(MobiHoc)*, 2003.

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

[5] L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, 24(11), November 1981.

[6] D. Liu and P. Ning. Establishing pairewise keys in distributed sensor networks. In *ACM Conference on Computer and Communication Security(CCS)*, 2003.

[7] D. Liu and P. Ning. Location-based pairewise key establishments for static sensor networks. In *ACM Workshop on Security in Ad Hoc and Sensor Networks(SASN)*, 2003.

[8] D. Malan, M. Welsh, and M. Smith. A public infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *1st IEEE International conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. Spins: security protocols for sensor networks. In *ACM MOBICOM*, 2001.

[10] G. Tel. Introduction to distributed algorithm. In *Cambridge University Press*, 2000.

[11] X. Wang, G. Xing, Y. Zhang, C. Li, R. Pless, and C. Gill. Integrated coverage and connectivity confi guration in wireless sensor networks. In *International Conference on Embedded Networked Sensor Systems(Sensys)*, 2003.

[12] H. Yang and S. Lu. Commutative cipher based en-route fi ltering in wireless sensor networks. In *IEEE VTC Wireless Security Symposium*, 2004.

[13] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route fi ltering of injected false data in sensor networks. In *INFOCOM*, 2004.

[14] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sens or networks. *ACM Wireless Networks(WINET)*, 11(2), March 2005.

[15] S. Zhu, S. Setia, and S. Jajodia. Leap: Effi cient security mechanisms for large-scale distributed sensor networks. In *ACM Conference on Computer and Communications Security(CCS)*, 2003.

[16] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authetication scheme for fi ltering false data in sensor networks. In *IEEE symposium SP*, 2004.