A Framework for Identifying Compromised Nodes in Sensor Networks

Qing Zhang Ting Yu Peng Ning
Department of Computer Science
North Carolina State University
{qzhang4, tyu, pning}@ncsu.edu

Abstract

Sensor networks are often subject to physical attacks. Once a node's cryptographic key is compromised, an attacker may completely impersonate it, and introduce arbitrary false information into the network. Basic cryptographic security mechanisms are often not effective in this situation. Most techniques to address this problem focus on detecting and tolerating false information introduced by compromised nodes. They cannot pinpoint exactly where the false information is introduced and who is responsible for it. We still lack effective techniques to accurately identify compromised nodes so that they can be excluded from a sensor network once and for all.

In this paper, we propose an application-independent framework for identifying compromised sensor nodes. The framework provides an appropriate abstraction of application-specific detection mechanisms, and models the unique properties of sensor networks. Based on the framework, we develop alert reasoning algorithms to identify compromised nodes. The algorithm assumes that compromised nodes may collude at will. We show that our algorithm is optimal in the sense that it identifies the largest number of compromised nodes without introducing false positives. We evaluate the effectiveness of the designed algorithm through comprehensive experiments.

1 Introduction

Compared with traditional wired and wireless networks, low-power wireless sensor networks can be rapidly deployed in a large geographical area in a self-configured manner. This makes them very suitable for real-time, large-scale information collection and event monitoring in hostile environments. Such applications, meanwhile, impose unique security challenges. In particular, since sensors are often deployed in open environments, they are vulnerable to physical attacks. Once recovering the keying materials of some nodes, an adversary is able to impersonate them completely, and inject arbitrary false information. Basic cryptographic security mechanisms, such as authentication and integrity protection, are often not effective against such impersonation attacks [8].

Recently, several approaches have been proposed to cope with compromised nodes in sensor networks, which mainly fall into two categories. Approaches in the first category are to detect and tolerate false information introduced by attackers [6, 16], in particular during data aggregation. Once the base station receives aggregated data, it checks their validity through mechanisms such as sampling and redundant sensors. These techniques, however, cannot identify exactly where the false information is introduced and who is responsible for it.

Approaches in the second category rely on application specific detection mechanisms which enable sensor nodes to monitor the activities of others nearby. Once observing an abnormal activity, a node may raise an alert either to the base station or to other nodes, who further determine which nodes are compromised. We call approaches in this category *alert-based*. Typical alert-based approaches include those in sensor network routing [9] and localization [13].

Such alerts make it possible to pinpoint compromised nodes. But how to effectively utilize such information is challenging. It is hard to decide whether to trust an alert since compromised nodes may as well raise false alerts to mislead the base station. Compromised nodes may further form a local majority and collude to increase their influences in the network. Moreover, existing alert-based approaches are application specific. They cannot be easily extended to other domains. A general solution to the *accurate identification* of compromised nodes still remains elusive.

The problem of identifying compromised nodes shares certain similarity with fault diagnosis in diagnosable systems [15]. However, in those systems, faults are assumed to be permanent, which means a faulty node will always fail a test, and thus can always be identified by fault-free nodes. Some later works relax permanent faults to intermittent faults [3], which however still assume that a faulty node cannot pass a test following certain probabilities. This

1-4244-0423-1/06/$20.00 ©2006 IEEE

assumption does not hold in sensor networks, where a compromised node may behave arbitrarily. For example, it may always report correct sensing data, and meanwhile issue false alerts. Such malicious behavior cannot be observed by an uncompromised node. Thus, we cannot directly apply works in self-diagnosable systems to identify compromised nodes in sensor networks.

The problem of false alerts (or feedback) and collusion also arises in decentralized systems such as online auction communities and P2P systems [10, 17]. One seemingly attractive approach is to apply existing trust management techniques in sensor networks. For example, we may identify sensor nodes with the lowest trust values as compromised. However, as a decentralized system, sensor networks bear some quite unique properties. Many assumptions in reputation-based trust management are not valid in sensor networks. Simply applying those techniques is unlikely to be effective (see section 4 for a detailed experimental comparison).

For example, in P2P systems interactions may happen between any two entities. If an entity provides poor services, it is likely that some other entities will be able to detect it and issue negative feedback accordingly. The interactions among sensor nodes, however, are restricted by the deployment of a sensor network. For a given node, only a fixed set of nodes are able to observe it. Thus, it is easy for compromised nodes to form local majorities.

Also, most decentralized systems are composed of autonomous entities, which pursue to maximize their own interests. Incentive mechanisms are needed to encourage entities to issue (or at least not discourage them from issuing) feedback. A sensor network, however, is essentially a distributed system. Sensor nodes are designed to cooperate and finish a common task. Thus, it is possible to design identification mechanisms that achieve *global* optimality for a given goal. For instance, to cope with false alerts, we may choose to identify as compromised both the target and the issuer of an alert, as long as it will improve the security of the whole system. Such an approach is usually not acceptable in P2P systems and online auction.

Indeed, the unique properties of sensor networks bring both challenges and opportunities. How to accommodate and take advantages of these properties is the key to the accurate identification of compromised nodes.

In this paper, we propose novel techniques to provide general solutions to the above problem. Our techniques are based on an application-independent framework that abstracts some of the unique and intrinsic properties of sensor networks. It can be used to model a large range of existing sensor network applications. In summary, our contributions include the following:

1. We generalize the model of alert-based approaches, and propose an application-independent framework for identifying compromised nodes. The central component of the framework is an abstraction of the monitoring relationship between sensor nodes. Such relationship can be derived from application specific detection mechanisms. The framework further models sensor nodes' sensing and monitoring capabilities, and their impacts on detection accuracy. This framework is built on detection mechanisms provided by applications. It does not require sensor nodes to support additional functionalities.

2. Based on the proposed framework, we design alert reasoning algorithms to accurately identify compromised sensor nodes. The algorithm does not rely on any assumptions on how compromised nodes behave and collude. We show that the algorithm is optimal, in the sense that given any set of alerts, our algorithm identifies the largest number of compromised nodes that can generate these alerts, without introducing any false positives. We also study how to tradeoff certain false positives to further eliminate compromised nodes.

3. We conduct comprehensive experiments to evaluate the proposed algorithm. The results show that it yields high detection rates with bounded false positive rates, and thus is effective in identifying compromised nodes in sensor networks.

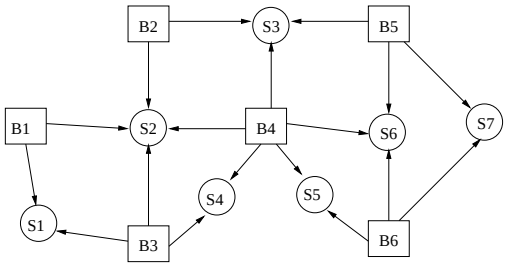
The rest of the paper is organized as follows. Section 2 presents a general framework for identifying compromised nodes, and shows how sensor network application can be modelled by the framework. In section 3, we propose an optimal algorithm to identify compromised sensor nodes. In section 4, we show the effectiveness of our algorithms through experimental evaluation. Section 5 reports closely related work. Concluding remarks are given in section 6.

2 A General Framework for Identifying Compromised Nodes

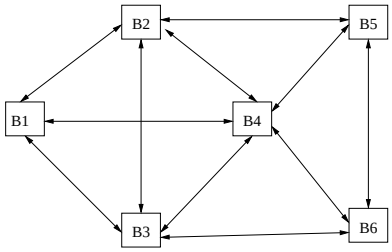
We first use an example to identify the aspects of sensor networks that are relevant to the identification of compromised nodes. We then present the general framework.

2.1 An Example Sensor Network Application

Many sensor network applications require sensors' location information (e.g., in target tracking). Since it is often too expensive to equip localization devices such as GPS receivers on every node, many location discovery algorithms depend on beacon nodes, i.e., sensor nodes aware of their locations. A non-beacon node queries beacon nodes nearby for references, and estimates its own location. An example



(a) The deployment of beacon nodes in sensor network localization



(b) The observability graph of sensor network localization

Figure 1. An example deployment of sensor nodes in sensor network localization and its corresponding observability graphs

deployment of the sensor network is shown in figure 1(a), where beacon nodes and non-beacon nodes are represented by rectangle and round nodes respectively. An edge from a beacon node b to a non-beacon node s indicates that b can provide location references to s.

A compromised beacon node may claim its own location arbitrarily, making non-beacon nodes around derive their locations incorrectly. Liu et al. [13] proposed a mechanism to detect malicious beacon nodes. The basic idea is to let beacon nodes probe each other and check the sanity of the claimed locations. If a beacon b_1 detects b_2 providing bogus location information, it will send alerts to the base station. Clearly, a compromised beacon node may also send false alerts to the base station.

After receiving a set of alerts from beacon nodes, what information is needed for the base station to make a rational decision on compromised nodes? First, the base station need the monitoring relationship to know whether an alert is valid, i.e., whether beacon nodes b_1 and b_2 are close enough to probe each other. Second, due to the imprecision of distance measuring, an uncompromised beacon node may raise alerts against another uncompromised one. The base station has to take this possibility into consideration. Third, it is necessary to regularly probe a beacon node so that it can be detected promptly if that beacon node is compromised and provides misleading location references.

The above information is quite relevant to reason about compromised nodes, and is commonly available in sensor network applications. It should be included in a general framework for identifying compromised nodes.

2.2 Assumptions

We make the following assumptions about sensor network applications before presenting a general framework for identifying compromised nodes.

First, we assume there exist application-specific detection mechanisms deployed in a sensor network, which enable sensor nodes to observe each other's behavior. Such detection mechanisms are commonly employed in sensor networks. Examples include beacon node probing in sensor network localization as mentioned above, witnesses in data aggregation [6] and the watchdog mechanism [9]. A sensor node s_1 is called an *observer* of another node s_2 if s_1 can observe the behavior of s_2. A node may have multiple observers or observe multiple other nodes.

Second, we focus on static sensor networks, where sensor nodes do not change their locations dramatically once deployed. In practice, many sensor networks are static. It implies that the observability relationship between sensor nodes does not change unless a sensor network is reconfigured.

Third, we assume that message confidentiality and integrity are protected through key management and authentication mechanisms [7], so that a sensor node can send information securely and reliably to the base station. Several techniques have been proposed in the literature to ensure the availability of such channels [2, 5].

Finally, we assume the base station is trusted, and has sufficient computation and communication capabilities. Hence, we adopt a centralized approach, where the base station is responsible for reasoning about alerts and identifying compromised nodes. The responsibility of each node is only to observe abnormal behavior and raise alerts to the base station as required by application specific detection mechanisms.

2.3 The Framework

With the above assumptions, our framework for identifying compromised nodes is composed of the following components:

Observability graph. An observability graph is a directed graph $G(V, E)$, where V is a set of vertices that represent sensor nodes, and E is a set of edges. An edge $(s_1, s_2) \in E$ if and only if s_1 is an observer of s_2. An observability graph is derived from the detection mechanism

of an application. Note that V only contains those nodes whose security is concerned by a detection mechanism. For example, in sensor network localization, the observability graph only includes beacon nodes. Figure 1(b) shows the observability graph corresponding to the sensor network localization deployment in figure 1(a).

Alerts. An alert takes the form (t, s_1, s_2), indicating that node s_1 observes an abnormal activity of s_2 at time t. Alerts may not need to be explicitly sent by sensor nodes. Instead, in some applications they may be implicitly inferred by the base station from application data sent by sensor nodes.

Sensor behavior model. Sensors are not perfect. Even if a node is not compromised, it may still occasionally report inaccurate information or behave abnormally. A sensor behavior model includes a parameter r_m (called the *reliability* of sensors) which represents the percentage of normal activities conducted by an uncompromised node. For example, in sensor network localization, $r_m = 0.99$ means 99% of the time an uncompromised beacon node provides correct location references.

Observer model. Similarly, an observer model represents the effectiveness of the detection mechanism of a sensor network, which is captured by its *observability rate* r_b, *positive accuracy* r_p and *negative accuracy* r_n. r_b is the probability that an observer s_1 observes an activity when it is conducted by an observee s_2. This reflects the fact that in some applications, due to cost and energy concerns, s_1 may not observe every activity of s_2. The positive accuracy r_p is the probability that s_1 raises an alert when s_2 conducts an abnormal activity observed by s_1. Similarly, r_n is the probability that s_1 does not raise an alert when s_2 conducts a normal activity observed by s_1. r_p and r_n reflect the intrinsic capability of a detection mechanism.

Security estimation. If it is possible that all the nodes in the network are compromised, then the base station cannot identify definitely which nodes are compromised based on alerts. Therefore, this framework focuses on the situation where the number of compromised nodes does not exceed a certain threshold K. We call K the *security estimation* of a network. How to determine K is application specific, depending on, e.g., the assumption of attackers' capability, the strength of sensors' keys, and how long the network has been deployed.

Identification function. An identification function F determines which nodes are compromised. Formally, it takes as inputs the observability graph G, the sensor reliability r_m, the observer model (r_b, r_p, r_n), the security estimation K, and a set of alerts raised during a period T, and returns a set of node IDs, which indicate those nodes that are considered compromised.

The above framework is application independent. It can be used to model a large range of sensor networks. We emphasize that our framework is built on the alert-based detection mechanisms provided by applications. The framework itself does not require sensor nodes to support additional functionalities, and thus does not introduce additional communication and computation costs to the network.

3 Identification of Compromised Nodes

In this section, we present our approach to trust function design based on the above framework.

Let s_1 be an observer of s_2. As sensors do not have perfect sensing and monitoring capabilities, even if an alert from s_1 against s_2 is observed, the base station cannot draw any definite conclusion from this single alert. Instead, it needs to observe the alert pattern during a certain period of time to discover suspicious activities with high confidence.

Given a network's sensor behavior model, its observer model and the frequency of the monitored events, we can derive the expected number of alerts raised by s_1 against s_2 during a period of time, when they are uncompromised. During the operation of the sensor network, the base station compares the number of alerts actually raised by s_1 against s_2 with the expected number. Only when the former is higher than the latter with statistical significance, should the base station consider it as abnormal. Specifically, for every event sensed by s_j, the probability that s_i raises an alert against s_j is given by $C = r_b \cdot ((r_m \cdot (1 - r_n) + (1 - r_m) \cdot r_p)$. Let $f_j(x)$ be the distribution of the number of events that can be sensed by s_j. Then the distribution of the number of alerts along the edge (s_i, s_j) (i.e., those raised by s_i against s_j) is $f_{ij}(x) = f_j(\frac{x}{C})$. For a period t, the expected number of alerts along (s_i, s_j) will be $R_{ij}(t) = C \int^t f_{ij}(x)$.

During a time period T, if the number of alerts along the edge (s_i, s_j) is over $R_{ij}(T) + \delta$, where $\delta \geq 0$ is an application specific parameter, then we say the edge (s_i, s_j) is an *abnormal edge* in the observability graph. Otherwise, (s_i, s_j) is *normal*. An abnormal edge can be interpreted as a definite claim from s_i that s_j is compromised. Similarly, a normal edge represents s_i's endorsement that s_j is not compromised. Note that it is possible that s_j is not compromised but malfunctioned. But in this case, we treat s_j as compromised anyway since information or services from s_j cannot be trusted anymore.

Given an abnormal edge (s_i, s_j), either s_i is compromised and raises many bogus alerts against s_j, or s_j is compromised and its malicious activities are observed by s_i, or both. If we further suppose there is an additional normal edge (s_l, s_j), then one of s_i and s_l must be compromised. Otherwise, the two edges should be consistent with each other since s_l and s_i observe the activities of the same node s_j during the same period of time.

Definition 3.1 *Given an observability graph $G(V, E)$, let*

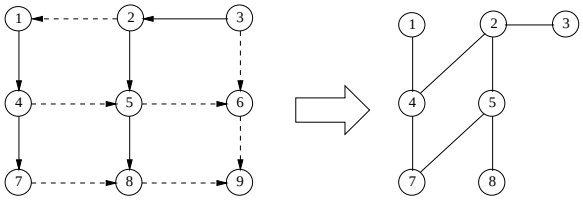


Figure 2. An observability graph and its corresponding inferred graph

E_a and E_n be the set of abnormal edges and normal edges in G respectively. We say that two sensor nodes s_i and s_j form a suspicious pair *if one of the following holds:*

1. *$(s_i, s_j) \in E_a$ or $(s_j, s_i) \in E_a$;*
2. *There exists a sensor node s', such that either $(s_i, s') \in E_a$ and $(s_j, s') \in E_n$, or $(s_i, s') \in E_n$ and $(s_j, s') \in E_a$.*

Let $\{s_1, s'_1\}, ..., \{s_k, s'_k\}$ be the suspicious pairs derived from an observability graph G. The inferred graph *of G is an undirected graph $I(V', E')$ such that $V' = \bigcup_{1 \le i \le k} \{s_i, s'_i\}$ and $E' = \{\{s_i, s'_i\} \mid 1 \le i \le k\}$.*

Intuitively, if (s_i, s_j) are a suspicious pair, then at least one of them is compromised. Note that if a pair of node is not suspicious, it does not mean that they are both uncompromised. It only means we cannot infer anything about them.

The left part of figure 2 shows an observability graph, where abnormal and normal edges are represented by solid and dashed edges respectively. The corresponding inferred graph is shown in the right part of the figure. Note that an inferred graph may not be connected in general.

From the figure one may wonder that, since $(2, 1)$ is normal and $(2, 5)$ is abnormal, shouldn't $\{1, 5\}$ be a suspicious pair too? The answer is no since it is possible that node 2 is compromised, and selectively issues bogus alerts against node 1 but not node 5, even though both node 1 and 5 are uncompromised. Similarly, a compromised node may sense data normally but issue bogus alerts or vise versa. So $\{1, 3\}$ is not a suspicious pair even though $(3, 2)$ is abnormal and $(2, 1)$ is normal. In other words, transitivity does not hold when constructing suspicious pairs.

Clearly, if a sensor node does not appear in the inferred graph, then its behavior is consistent with the sensor behavior model and observer model, and thus should be considered uncompromised. Hence, we concentrate on identifying compromised sensor nodes among those involved in the inferred graph.

Definition 3.2 *Given an inferred graph $I(V, E)$ and a security estimation K, a* valid assignment *with regard to I and K is a pair (S_g, S_b), where S_g and S_b are two sets of sensor nodes that satisfies all of the following conditions:*

1. *S_g and S_b is a partition of V, i.e., $S_g \cup S_b = V$ and $S_g \cap S_b = \emptyset$;*
2. *For any two sensor nodes s_i and s_j, if $s_i \in S_g$ and $s_j \in S_g$, then $\{s_i, s_j\} \notin E$; and*
3. *$|S_b| \le K$.*

Intuitively, a valid assignment corresponds to one possible way that sensor nodes are compromised, that is, when they raise false alerts and conduct abnormal activities, the resulting inferred graph is I. S_g and S_b contains the uncompromised and compromised nodes respectively. For a given inferred graph and a security estimation K, there may exist many valid assignments. Obviously the common nodes in all possible assignments are always compromised, and others may or may not be compromised, depending on which assignment is true for the actual system. This inspires us that an optimal algorithm is to identify the common nodes in all possible assignments, thus it will identify the largest number of truly compromised nodes, and does not introduce any false positives.

Definition 3.3 *Given an inferred graph $I(V, E)$ and a security estimation K, let $\{(S_{g1}, S_{b1}), \ldots, (S_{gn}, S_{bn})\}$ be the set of all the valid assignments with regard to I and K. We call $\bigcap_{1 \le i \le n} S_{bi}$ the* compromised core *of the inferred graph I with security estimation K, denoted $CompromisedCore(I, K)$. Similarly, $\bigcap_{1 \le i \le n} S_{gi}$ is called the* uncompromised core *of I with security estimation K, denoted $UncompromisedCore(I, K)$.*

Definition 3.4 *Let I be the inferred graph, given an observability graph G, a sensor behavior model, an observer model, a security estimation K, and a set of alerts during a time period T. We say an identification function is F* optimal *if and only if F always returns $CompromisedCore(I, K)$.*

Given the general framework, *one key problem is thus to develop algorithms that efficiently compute $CompromisedCore(I, K)$.* On the other hand, though introducing no false positives, only identifying the compromised core may not achieve high detection rates since there may still exist many suspicious pairs whose nodes are not included in the compromised core. Thus *another key problem is to seek techniques that further eliminate compromised nodes without causing many false positives.*

In summary, our approach is composed of two phases. In the first phase, we compute or approximate the compromised core, identifying those nodes that are definitely compromised. In the second phase, we tradeoff accuracy for eliminating more compromised nodes.

3.1 The Algorithm to Identify Compromised Sensor Nodes

Though collusion between compromised nodes is good for an attacker, an identification function should not rely on any assumptions of collusion models. Otherwise, an attacker may easily defeat the identification algorithm by slightly changing the behavior of compromised nodes and making the collusion assumption invalid. For example, even if s_1 issues a lot of alerts against s_2, we cannot conclude that one of them is compromised and the other is not. It is possible that both of them are compromised and the attacker just wants to confuse the identification function.

On the other hand, no matter how compromised nodes collude, it always holds that a suspicious pair contains at least one compromised node. This property helps us derive the lower bound of the number of compromised nodes.

Lemma 3.1 *Given an inferred graph $I(V, E)$, let V_I be a minimum vertex cover of I. Then the number of compromised nodes is no less than $|V_I|$.*

We denote the size of the minimum vertex covers of an undirected graph G as C_G. Given a sensor node s, the neighbors of s in an inferred graph I is denoted $\mathcal{N}_s$. Further, let I'_s denote the graph after removing s and its neighbors from I. We have the following theorem for identifying compromised sensor nodes.

Theorem 3.1 *Given an inferred graph I and a security estimation K, for any node s in I, $s \in CompromisedCore(I, K)$ if and only if $|\mathcal{N}_s| + C_{I'_s} > K$.*

Intuitively, if we assume a sensor node s is uncompromised, then all its neighbors in I must be compromised. According to lemma 3.1, there are at least $|\mathcal{N}_s| + C_{I'_s}$ compromised nodes, which should be no more than the security estimation K. Otherwise, s must be compromised. Meanwhile, if $|\mathcal{N}_s| + C_{I'_s} \leq K$, we can always construct a valid assignment for I with regard to K, where s is assigned as uncompromised, which means s is not in $CompromsedCore(I, K)$.

By theorem 3.1, the algorithm to identify $CompromisedCore(I, K)$ is straightforward. For each node s, we check whether $|\mathcal{N}_s| + C_{I'_s}$ is larger than K. Unfortunately, this algorithm is in general not efficient since the minimum vertex covering problem is NP-complete. In theory we also have to compute the minimum vertex cover of a different graph when checking each node.

Thus, we seek efficient algorithms to approximate the size of minimum vertex covers. To prevent false positives, we are interested in deriving a good *lower bound* of the size of minimum vertex covers, a goal different from that of many existing approximation algorithms. In this paper, we choose the size of maximum matchings of I as such an approximate. We denote the size of the maximum matchings of an undirected graph G as M_G.

```
AppCompromisedCore(I, K)
//Input: I is an inferred graph
// K is a security estimation
//Output: the compromised core of I with K
S_b = ∅
For each sensor node s in I
    Let n_s be the number of neighbors of s
    Let m = M_{I'_s}
    If n_s + m > K
        S_b = S_b ∪ {s}
Return S_b
```

Figure 3. An Efficient algorithm to approximate the compromised core

Lemma 3.2 *Given an undirected graph G, $M_G \leq C_G \leq 2M_G$. And the bounds are tight.*

Corollary 3.1 *Given an inferred graph I and a security estimation K, for any node s in I, if $|\mathcal{N}_s| + M_{I'_s} > K$, then $s \in CompromisedCore(I, K)$.*

A maximum matching of an undirected graph can be computed in polynomial time [14]. Figure 3 shows an efficient algorithm to approximate the compromised core. Since this algorithm does not assume any specific collusion model among compromised nodes, we call it the *general identification algorithm*.

Theorem 3.2 *The complexity of the algorithm* AppCompromisedCore *is $O(mn\sqrt{n})$, where m is the number of edges and n is the number of vertices in an inferred graph [14].*

3.2 Further Elimination of Compromised Sensor Nodes

The above algorithm does not introduce any false positives. Compromised nodes identified by the above algorithms may be safely excluded from the networks through, e.g., key revocation [7, 11]. However, there may still be suspicious pairs left that do no include any nodes in the compromised core. We call the graph composed of such pairs the *residual graph*.

We may tradeoff accuracy for eliminating more compromised nodes. Since a suspicious pair contains at least one compromised node, identifying both nodes as compromised will introduce at most one false positive. By computing the maximum matching of a residual graph and treating them as compromised, the false positive rate is bounded by 0.5. Note that this is the *best* we can do only based on a given

set of alerts. In order to reduce this worst-case false positive rate, application specific information or assumptions are needed.

In summary, given an inferred graph and a security estimation, our approach is first to approximate its compromised core. We then compute the maximum matching of the residual graph, and further eliminate compromised nodes.

4 Experiments

We simulate a sensor network deployed to monitor the temperature of an area of $100m \times 100m$. For simplicity, we assume sensor nodes are randomly distributed in the area. We adopt a simple detection mechanism. If the distance between two sensor nodes is within 10 meters, and the temperatures reported by them differ by more than $1\,^{\circ}C$, the base station infers that each of them raises an alert against the other. In other words, two nodes are observers of each other if they are with in 10 meters.

Sensor nodes report temperatures to the base station once per minute, and the sensed data follows a normal distribution $N(\mu, \sigma^2)$, where μ is the true temperature at a location, and $\sigma = 0.2$, which is consistent with the accuracy of typical thermistors in sensor network [1].

Unless otherwise stated, we assume $10\% \sim 15\%$ of the nodes in the network are compromised. The security estimation $K = 0.15N$, where N is the total number of nodes in the network. The goal of the attacker is to raise the temperature reading in the area. The attacker may choose to either randomly compromise nodes in the field, in which case no local majority is formed, or compromise nodes centered around a position (x, y), following a normal distribution with a standard deviation σ_d. The latter corresponds to the case of local majority, and the parameter σ_d controls its strength. The smaller σ_d is, the closer compromised nodes are to each other, and thus the stronger the local majority is. We call σ_d the *concentration* of compromised nodes.

The evaluation metrics of the experiments are the detection rates and false positive rates of the proposed identification approach.

We act conservatively, and assume that compromised nodes are in fact colluding, but the base station does not know this and cannot utilize any knowledge of the collusion. Compromised nodes all report the same false temperature so that there are no alerts between them. We evaluate the effectiveness of the general *AppCompromisedCore* algorithm followed by the maximum matching approach. We call it *general+mm* in the experiment.

We further compare our approach with a simple voting mechanism: if the number of a node s's neighbors who raise alerts against s is more than those who do not, then s is considered compromised.

We also compare our approach with EigenTrust [10] and PeerTrust [17], two well-known reputation-based trust functions for P2P systems. EigenTrust and PeerTrust both derive a unique global trust value for each entity from that entity's transaction records. Applied in sensor networks, the global trust values can be used to rank sensor nodes, where those with low trust values are identified as compromised. Therefore, they can be compared with general+mm.

We conduct three sets of experiments to evaluate the impacts of the following factors on the effectiveness of the above approaches.

4.1 Local Majority

Figure 4 shows the effectiveness of different approaches when compromised nodes form local majorities. The number of nodes in the network is set to be 200. The concentration of compromised nodes is varied from 5 to 100. When compromised nodes are extremely close to each other, they essentially form a cluster. Suspicious pairs only involve those compromised nodes near the edge of the cluster. Those near the center of the cluster do not appear in the inferred graph, and thus cannot be identified. We note that even in this situation the general+mm approach can still achieve detection rate over 0.6, mainly due to the maximum matching approach in the second phase. When the compromised nodes are less concentrated, the general identification algorithm enables the base station to quickly identify almost all the compromised nodes. That is why we see a quick rise in general+mm's detection rate and a sharp drop in its false positive rate.

When compared with other schemes, we have the following observations. First, EigenTrust seems to be inferior to general+mm and PeerTrust. The reason is that EigenTrust relies on the existence of pre-trusted peers to identify malicious collectives, which correspond to colluding compromised nodes in our setting. Without pre-trusted peers, it cannot significantly distinguish malicious entities from good ones. That is why we see an upper bound of the detection rate of EigenTrust even when compromised nodes do not form a strong local majority.

Second, we notice that when the concentration is over 20, PeerTrust and voting mechanism actually yield comparable detection rate to that of general+mm with a little bit lower false positive rates. A closer examination of the network reveals that, with 200 nodes in the network, the average number of observers for each node is around 3. When the concentration is 20, among the neighbors of a compromised node, on the average no more than 1 neighbor is compromised. In other words, when the concentration is over 20, compromised nodes seldom form local majorities. In this case PeerTrust and simple voting, both relying on majority voting mechanisms, are more likely to assign low

trust values to compromised nodes or label them as compromised nodes directly. For general+mm, each identified compromised nodes in the second phase will result in the sacrifice of an uncompromised nodes, resulting in higher false positive rates.

Third, when the compromised nodes form strong local majorities (i.e., the concentration is smaller than 20), general+mm yields much higher detection rates and lower false positive rates than PeerTrust. And the simple voting has the poorest detection rate as low as 10%, as it does not do any reasoning on the credibility of the feedback. This is an important advantage of our approach. In sensor networks, it is always cost-effective for attackers to compromised a small portion of the network, and make them collude. Otherwise either they have to compromise a large portion of the network, which is very costly and often not feasible, or they do not collude, in which case any voting-based algorithm can identify most of the compromised nodes, as shown above. So it is important that an identification algorithm performs well even when local majorities are formed by compromised nodes. From the experiment we see when collusion is the strongest, although the false positive rate of our algorithm is close to 50%, it is still the lowest among all solutions, and also achieves the highest detection rate.

4.2 Sensor Node Density

In the next experiment, we vary the number of sensor nodes in the area from 50 to 200. As we have seen from previous experiment, when there is no collusion among compromised nodes, all algorithms have high detection rate and false positive rate, and our algorithm outperforms other 3 algorithms when there exists a strong collusion. Thus we report the experimental results which set the concentration of compromised nodes to be 15, a not too strong case. We have also tried other concentration parameters and observed similar trends. Figure 5(a) and 5(b) show respectively the detection rate and the false positive rate of each approach.

We see that when the number of sensor nodes increases, all the approaches achieve better detection rates and less false positive rates. Intuitively, the more densely sensor nodes are deployed, the more observers each node has, and thus the more likely an abnormal activity is detected. The second observation is that general+mm do not achieve high detection rates when sensor nodes are deployed very loosely. This is because in this situation many nodes do not have any observers. There are too few alerts for the base station to identify compromised nodes definitely. The third observation, we see that general+mm detects much more compromised nodes with similar false positives than all other approaches, due to the fact that it takes the unique properties of sensor networks into consideration so that it is more resilient to compromised nodes forming local majorities.

4.3 Accuracy of The Security Estimation

The security estimation gives an upper bound of the number of compromised nodes. It is an important parameter for identification functions to infer compromised nodes. An accurate security estimation is not expected to always be available. The next experiment evaluates how the accuracy of a security estimation affects the effectiveness of different approaches. The total number of sensor nodes are set to be 200. The number of compromised nodes is 20, and their concentration is set to be 15. The security estimation is varied from 20 to 40. Voting mechanism is not evaluated in this experiment as it does not involve this parameter when identifying compromised nodes. The experiment results are shown in figure 6.

We see that general+mm still achieve very high detection rates even when the accuracy of the security estimation decreases. When the security estimation is accurate, most of the compromised nodes are identified by the algorithm in the first phase, producing few false positives. When the accuracy of the security estimation decreases, the effectiveness of the first phase also decreases. More compromised nodes are instead identified in the second phase by the maximum matching approach. That explains why the false positives increase while detection rates remain high. The detection rates of EigenTrust and PeerTrust in fact improve a little bit when the security estimation accuracy decreases. This is because they always identify the K nodes with the lowest trust values as compromised, which will include more compromised nodes as K increases. But this also increases their false positive rates.

In the next experiment, we evaluate the impact on the effectiveness of our algorithms when the security estimation is in fact less than the actual number of compromised nodes in a network. Same as the previous experiment, we set the number of compromised nodes to be 20 and vary the security estimation K from 1 to 19. We observe that our algorithm always throws an exception, stating the size of the minimum vertex cover of the inferred graph is greater than K. This exception indicates that we underestimate the number of compromised nodes when setting K because it contradicts with lemma 3.1. We realize that such underestimation may not always be detected. It is possible that with certain observability graphs, an attacker might be able to compromise more than K sensor nodes, and submit false data and alerts in a way such that the size of the minimum vertex cover of the resulting inferred graph is no more than K. We will study the properties of such graphs in future work. Currently, we treat the exceptions as chances to use multiple K for the detection. That is, if there is an exception, then we should go back and double check our estimation about K, then recompute another bigger K and run our algorithm again.

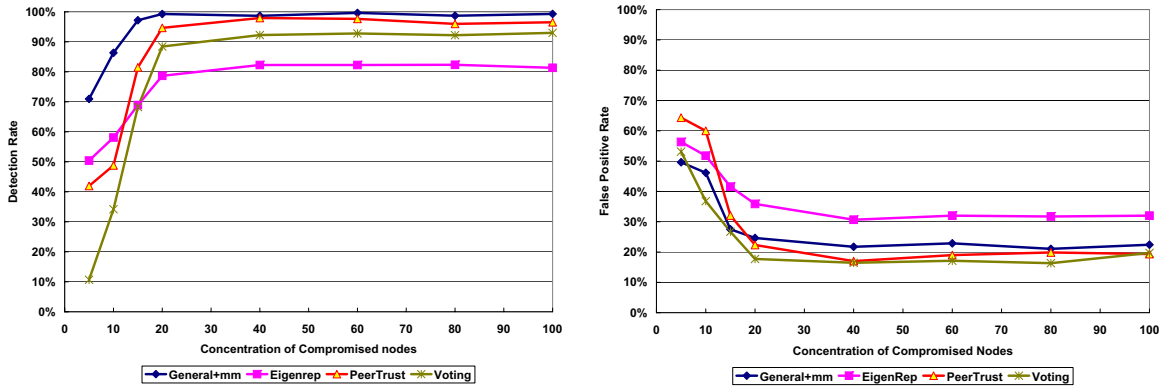


Figure 4. The impact of the concentration of compromised sensor nodes

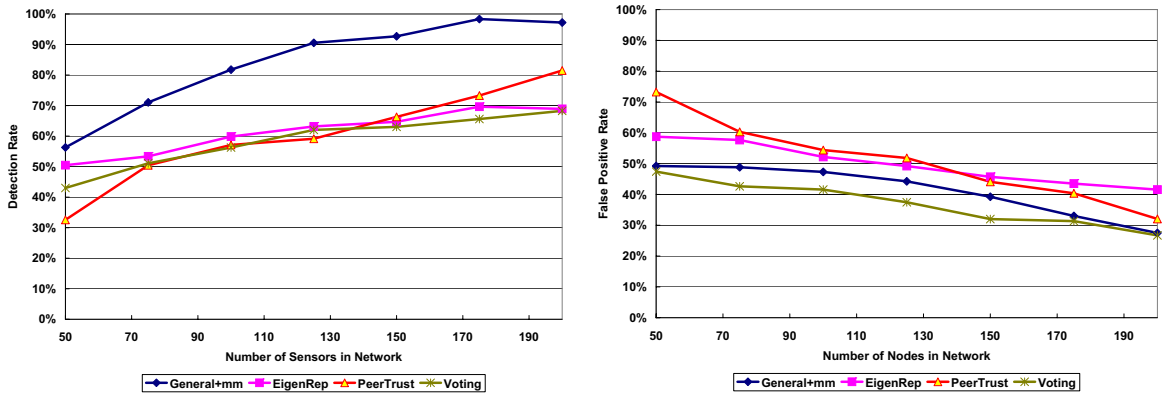


Figure 5. The impact of the deployment density of sensor nodes

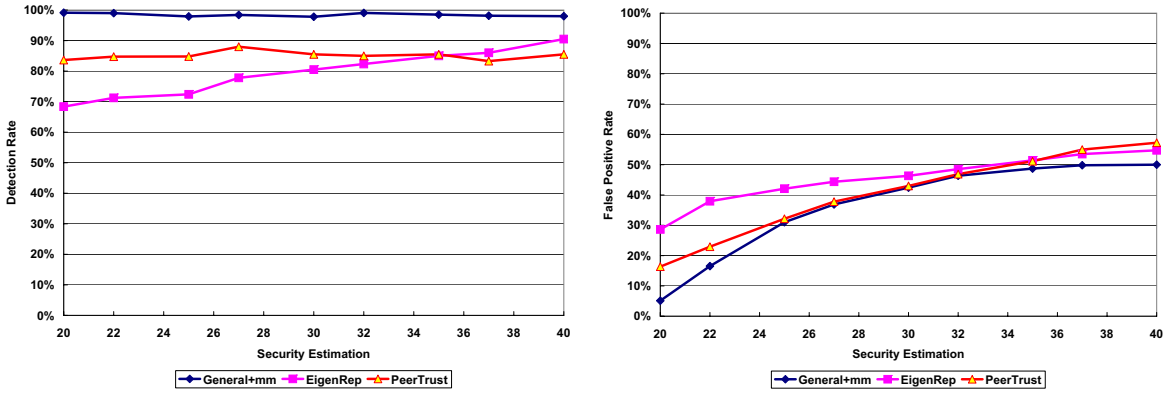


Figure 6. The impact of the accuracy of security estimation

5 Related Work

Much work has been done to provide security primitives for wireless sensor networks, including practical key management [7, 11], broadcast authentication [12], and data authentication [16] as well as secure in-network processing [4]. The work of this paper is complementary to the above techniques, and can be combined to achieve high information assurance for sensor network applications. Several approaches have been proposed to detect and tolerate false information from compromised sensor nodes [6, 16] through e.g., sampling and redundancy. But they do not pro-

vide mechanisms to accurately identify compromised sensor nodes, which is the focus of this paper.

Reputation-based trust management has been studied in different application contexts, including P2P systems, wireless ad hoc networks, social networks and the Semantic Web [10, 17]. Many trust inference schemes have been proposed. They different greatly in inference methodologies, complexity and accuracy. As discussed early, the interaction model and assumptions in the above applications are different from sensor networks. Directly applying existing trust inference schemes may not yield satisfactory results in sensor networks.

Ganeriwal et al. [9], propose to detect abnormal routers in sensor networks through reputation mechanism. Their decentralized trust inference approach shows the usefulness of reputation in sensor networks. But their approach treats a sensor network the same as a typical P2P system, and thus does not capture the unique properties of sensor networks. Further, their work focuses on avoiding services from potentially compromised sensors instead of identifying and excluding them from sensor networks. Further, their work is application specific, and cannot be easily applied to other sensor network applications.

6 Conclusion

In this paper, we present a general framework that abstracts the essential properties of sensor networks for the identification of compromised sensor nodes. The framework is application-independent, and thus can model a large range of sensor network applications. Based on the framework, we develop efficient algorithms that achieve maximum accuracy without introducing false positives. We further propose techniques to trade off accuracy for increasing the identification of compromised nodes. The effectiveness of these techniques are shown through theoretical analysis and detailed experiments. To the best of our knowledge, our work is the first in the field to provide an application-independent approach to identify compromised nodes in sensor networks.

In the future we plan to investigate light-weighted decentralized approaches, and systematically analyze its benefits and inherent weakness when compared with centralized approaches.

References

[1] *MTS/MDA Sensor and Data Acquisition Boards User Manual*, May 2003.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *ACM Wireless Networks*, 7(6):609–616, 2001.

[3] A. Dahbura, K. Sabnani, and L. King. The comparison approach to multiprocessor fault diagnosis. *IEEE Trans. on Computers*, C-36(3):373–378, 1987.

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

[5] J. Deng, R. Han, and S. Mishra. A Robust and Light-Weight Routing Mechanism for Wireless Sensor Networks. In *Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS)*, 2004.

[6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A Witness-Based Approach For Data Fusion Assurance In Wireless Sensor Networks. In *IEEE 2003 Global Communications Conference (GLOBECOM)*, 2003.

[7] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003.

[8] W. Du, L. Fang, and P. Ning. Lad: Localization anomaly detection for wireless sensor networks. In *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[9] S. Ganeriwal and M. B. Srivastava. Reputation-based Framework for High Integrity Sensor Networks . In *ACM Security for Ad-hoc and Sensor Networks (SASN 2004)*, 2004.

[10] S. Kamvar, M. Schlosser, and H. Garcia-Molina. EigenRep: Reputation Management in P2P Networks. In *Twelfth International World Wide Web Conference*, 2003.

[11] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *10th ACM conference on computer and communications security (CCS '03)*, 2003.

[12] D. Liu, P. Ning, and W. Du. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *10th Annual Network and Distributed System Security Symposium (NDSS'03)*, 2003.

[13] D. Liu, P. Ning, and W. Du. Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks . In *Proceedings of the The 25th International Conference on Distributed Computing Systems (ICDCS '05)*, 2005.

[14] S. Micali and V. Vazirani. An $o\sqrt{|V|}|e|$ algorithm for finding maximum matchings in general graphs. In *21st. Symp. Foundations of Computing*, 1980.

[15] F. P. Preparata, G. Metze, and R. T. Chien. On the connection assignment problem of diagosable systems. *IEEE Trans. on Electonic Computers*, 16(6):848–854, 1967.

[16] B. Przydatek, D. Song, , and A. Perrig. SIA: Secure information aggregation in sensor networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, 2003.

[17] L. Xiong and L. Liu. Building Trust in Decentralized Peer-to-Peer Electronic Communities. In *The 5th International Conference on Electronic Commerce Research. (ICECR)*, 2002.