A Framework for Identifying Compromised Nodes in Wireless Sensor Networks

QING ZHANG, TING YU, and PENG NING
North Carolina State University

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 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.

In this article, we propose an application-independent framework for accurately 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.

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

General Terms: Algorithms, Security

Additional Key Words and Phrases: Sensor networks, intrusion detection

ACM Reference Format:
Zhang, Q., Yu, T., and Ning, P. 2008. A framework for identifying compromised nodes in wireless sensor networks. ACM Trans. Inf. Syst. Secur. 11, 3, Article 12 (March 2008), 37 pages. DOI = 10.1145/1341731.1341733. http://doi.acm.org 10.1145/1341731.1341733.

Based on "A Framework for Identifying Compromised Nodes in Wireless Sensor Networks" by Qing Zhang, Ting Yu, Peng Ning which appeared in *Proceedings of 2nd International Conference on Security and Privacy in Communication Networks (SecureComm)*, Baltimore, MD © 2006 IEEE.
The work of Zhang and Yu was partially supported by the National Science Foundation (NSF) under grants IIS-0430274. Ning's work was supported by the NSF under grants CAREER-0447761 and CNS-0430223.
Authors' address: Cyber Defense Laboratory, Department of Computer Science, North Carolina State University, Raleigh, NC 27695; emails: {qzhang4, yu, pning}@ncsu.edu.

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 particularly suitable for real-time, large-scale information collection and event monitoring for mission-critical applications in hostile environments, such as target tracking and battlefield surveillance.

Such applications, meanwhile, impose unique security challenges. Because sensors are extremely resource constrained, many existing security mechanisms cannot be directly applied to sensor networks [Perrig et al. 2001]. In recent years, we have witnessed great efforts toward designing security primitives specific for wireless sensor networks. For example, a variety of authentication and key management schemes have been proposed and implemented [Camtepe and Yener 2004; Liu et al. 2005]. On the other hand, because 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 mechanisms, such as authentication and integrity protection, are usually not effective against such impersonation attacks [Du et al. 2005].

Recently, several approaches have been proposed to cope with compromised nodes. These approaches mainly fall into two categories. Approaches in the first category are to detect and tolerate false information introduced by attackers [Du et al. 2003a; Hu and Evans 2003; Przydatek et al. 2003; Ye et al. 2004], in particular during data aggregation. Once the base station receives aggregated data, it checks their validity through mechanisms such as sampling and deployment of redundant sensors. However, these techniques often cannot be used to identify where the false information is introduced and who is responsible for it.

Approaches in the second category rely on application-specific detection mechanisms that enable sensor nodes to monitor the activities of others nearby. Once an abnormal activity is observed, 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. Representative alert based approaches include those in sensor network routing [Ganeriwal and Srivastava 2004] and localization [Liu et al. 2005].

Alerts from sensor nodes make it possible to pinpoint compromised nodes. However, how to effectively use such information is a very challenging problem. It is hard to decide whether an alert can be trusted because it is very likely that compromised nodes raise false alerts to mislead the base station and other nodes. Compromised nodes may further form a local majority in the network and collude, increasing their influences in the network. Further, existing alert-based approaches are specific to certain applications and cannot be easily extended to other domains. A general solution to the accurate identification of compromised nodes still remain elusive.

The problem of identifying compromised nodes shares certain similarity with fault diagnosis in diagnosable systems [Araki and Shibata 2003; Dahbura and Masson 1984; Fuhrman 1996; Preparata et al. 1967; Sullivan 1988]. 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 [Dahbura et al. 1987; Kozlowski and Krawczyk 1991], which however still assume that a faulty node cannot pass a test following certain probabilities. These assumptions do 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 from malicious entities also arises in other decentralized systems, such as online auction communities and P2P systems [Aberer and Despotovic 2001; Kamvar et al. 2003; Lee et al. 2003; Mui et al. 2002; Richardson et al. 2003; Xiong and Liu 2002; Yu and Singh 2002]. Reputation-based trust management has been adopted as an effective means to form cooperative groups in the above systems. 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 quite unique properties and significantly differ from the above systems. Many assumptions in reputation-based trust management do not hold in sensor networks. Thus, 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 misleading information or poor services, it is likely that some other entities will be able to detect it and issue negative feedback accordingly. The interactions between sensor nodes, however, are restricted by the deployment of a sensor network. For a given node, only a fixed set of nodes is able to observe it. Thus, it is easy for compromised nodes to form local majorities.

Also, most decentralized environments 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 about others. A sensor network, on the other hand, is essentially a distributed system, where all the sensor nodes are designed to cooperate and finish a common task. Therefore, 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 communities.

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

(1) We develop an application-independent framework for identifying compromised nodes based on alerts generated by specific detection mechanisms in sensor networks. 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. We show by example that many existing sensor networks can be easily modeled by the proposed framework. As our framework is built on the alert-based detection mechanisms provided by applications, it does not require sensor nodes to support additional functionalities, nor does it impose additional communication and computation costs to the network.

(2) Based on the proposed framework, we design an alert reasoning algorithm 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 trade off certain false positives to further eliminate compromised nodes.

(3) To better understand the capability of the above reasoning algorithm, we further consider a special case where all the compromised nodes actually collude with each other (i.e., they behave consistently and do not raise alerts against each other). We develop an identification algorithm when assuming the base station is aware of such collusion. The identification capability of this special algorithm serves as an upper bound for that of the general algorithm.

(4) We conduct comprehensive experiments to evaluate the proposed algorithm. The results show that it yields high detection rates and bounded false: positive rates and thus is effective in identifying compromised nodes.

The rest of the article is organized as follows. Section 2 presents a general framework for identifying compromised nodes and shows how sensor network application can be modeled by the framework. In Section 3, we present algorithms that identify compromised sensor nodes with optimal accuracy, for both the collusion and noncollusion cases. In Section 4, we show the effectiveness of our algorithms through experimental evaluation. Some relevant issues are

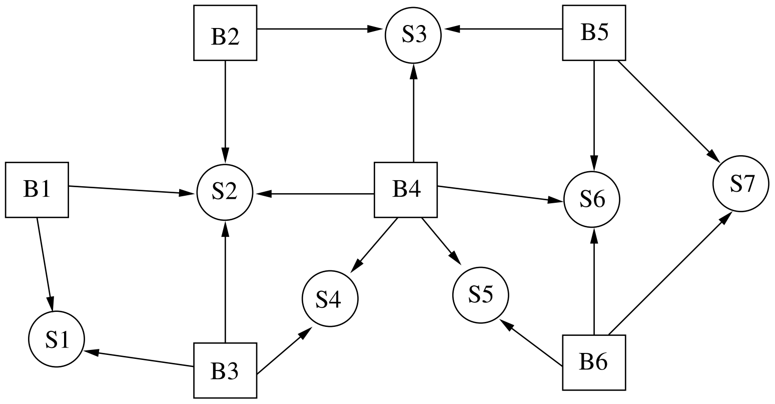


Fig. 1. The deployment of beacon nodes in sensor network localization.

discussed in Section 5. Section 6 reports closely related work to this paper. Concluding remarks are given in Section 7.

2. A GENERAL FRAMEWORK FOR IDENTIFYING COMPROMISED NODES

In this section, 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). Because it is often too expensive to equip localization devices such as GPS receivers on every node, many location discovery algorithms depend on beacon nodes, that is, sensor nodes aware of their locations. A nonbeacon node queries beacon nodes nearby for references and estimates its own location. An example deployment of the sensor network is shown in Figure 1, where beacon nodes and nonbeacon nodes are represented by square and round nodes, respectively. An edge from a beacon node b to a nonbeacon node s indicates that b can provide location references to s.

A compromised beacon node may claim its own location arbitrarily, making nonbeacon nodes around derive their locations incorrectly. Liu and colleagues [2005] 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. Suppose beacon node b_1's location is (x, y) and beacon node b_2 claims its location to be (x', y'). If the difference between the derived distance and the measured distance exceeds a threshold ϵ, then b_1 will consider b_2 as compromised and report 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 by the base station to make a rational decision on compromised nodes? First, the base station has to know whether an alert is valid, that is, whether beacon nodes b_1 and b_2 are close enough so that they can probe each other. In other words, the monitoring relationship between beacon nodes is needed.

Second, due to the imprecision of distance measuring, it is possible that an uncompromised beacon node raises an alert 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 for the base station to reason about compromised nodes and is commonly available in sensor network applications. Thus, it should be included by 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 that 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 [Du et al. 2003a], and the watchdog mechanism [Ganeriwal and Srivastava 2004]. 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.

The detection mechanisms are not assumed to be completely accurate. But we do assume that, given a sufficient number of observations of the same node, the number of alerts issued by two uncompromised observers should be statistically consistent. In other words, if a node's behavior during a time period T is abnormal, then all the uncompromised observers should raise a significantly high number of alerts during T. Whether the number of alerts is significantly high depends on the characteristics of the sensor node and the detection mechanism, which will be discussed in Section 2.3.

Second, we focus on static sensor networks, where sensor nodes do not change their locations dramatically once deployed. A large range of sensor networks fall into this category, for example, target tracking and environmental monitoring. One consequence of this assumption is 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 [Du et al. 2003b], 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 [Bose et al. 2001; Deng et al. 2004].

Finally, we assume the base station of a sensor network is trusted and has sufficient computation and communication capabilities. Hence, we adopt a centralized approach, where the base station is responsible for reasoning about the alerts and identifying compromised nodes. The responsibility of each node is only to observe abnormal activities and raise alerts to the base station. We will briefly discuss decentralized approaches in Section 5, where sensor nodes also take part in the reasoning and identification process.

2.3 The Framework

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

—*Observability graph.* An observabi 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. V contains only those nodes whose security is concerned and is involved in the underlying detection mechanism. For example, in the sensor network localization problem, the observability graph includes only beacon nodes.

—*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. The information in an alert may be further enriched, for example, by including s_1's confidence on the alert. For simplicity, we omit such parameters in this article. Note that alerts may not need to be explicitly sent by sensor nodes. Instead, in some applications they can be implicitly inferred by the base station from the sensing data sent by sensor nodes (e.g., when the reported data of two adjacent sensor nodes differ than a certain threshold).

—*Sensor behavior model.* Sensors are not perfect. Even if a node is uncompromised, it may still occasionally report inaccurate information or behave abnormally. A sensor behavior model includes a parameter r_m that represents the percentage of normal activities conducted by an uncompromised node. We call r_m the reliability of sensors. For example, in sensor network localization, if $r_m = 0.99$, then 99% of the time, an uncompromised beacon node provides the 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 $r_b(i, j)$, positive accuracy r_p, and negative accuracy r_n. Suppose s_1 is an observer of s_2. $r_b(1, 2)$ is the probability that s_1 observes an activity conducted by s_2. This reflects the fact that, in some applications, due to cost, energy concerns, and distance between nodes, s_1 may not be able to observe every activity of s_2. For simplicity, in this article we use a global r_b value between all pairs of nodes. 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.

The sensor behavior model and the observer model can usually be obtained from the specification of sensors and an application's detection mechanisms.

—*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, for example, the estimation of attackers' capability, the strength of sensors' keys, and how long the network has been deployed. We emphasize that K is only an upper bound of the number of compromised nodes. The base station does not need to know the exact number of compromised nodes in a network.

—*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.

We note 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.

Further, for simplicity, the above framework assumes that all the nodes and observers follow the same sensor behavior model and observer model. In some applications, there may be different types of sensor nodes with different sensing and observing capabilities. Our framework can be easily extended to model such applications by specifying different models for different types of sensor nodes.

With the above framework, there are two key problems: first, whether it is easy to model sensor network applications using the framework; and second, how to design efficient and accurate identification functions.

2.4 The Applicability of the Framework

The above framework is application independent and thus can be used to model a large range of sensor networks. In this section, we use two examples to show its applicability.

Example 1: Localization in Sensor Networks. The approach for detecting compromised beacon nodes [Liu et al. 2005] described in Section 2.1 can be modeled by our framework as follows.

(1) *Observability graph.* The vertices of the observability graph include only beacon nodes. There is a bidirectional edge between two beacon nodes, if they are close enough to probe each other. (Note that a detecting beacon node has to use a pseudoID to prevent a compromised beacon node from recognizing a probing query.) Figure 2 shows the observability graph corresponding to Figure 1.

(2) *Alerts.* If at time t a beacon node b_1 detects a bogus location claimed by a nearby beacon b_2, it will send an alert (t, b_1, b_2) to the base station.

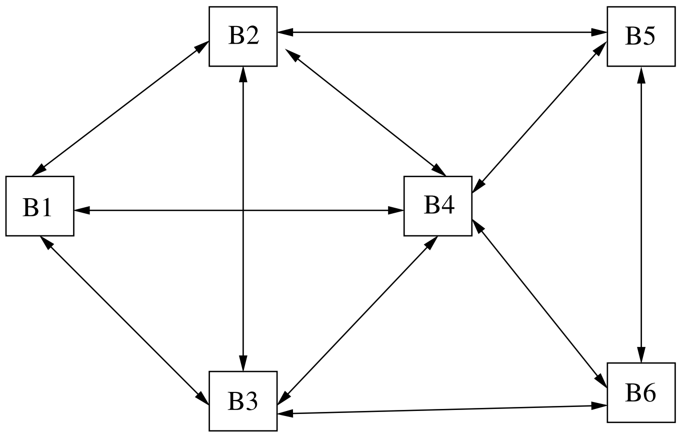


Fig. 2. The observability graph of sensor network localization.

(3) *Sensor behavior model.* The probability that an uncompromised beacon claims an incorrect location is determined by the resolution and the accuracy of the localization device. Let (x, y) be a beacon's actual location and (x', y') be its measured location. Assume $x' - x$ and $y' - y$ follow normal distribution $N(0, \sigma^2)$. Then the distribution of the distance d between (x, y) and (x', y') can be computed from bivariate transformation as $\frac{d}{2\pi^2\sigma^2}e^{-d^2/2\sigma^2}$. Suppose a beacon node is considered defective if d is larger than a predefined threshold ϵ. Then the reliability of beacon nodes is $r_m = P(d < \epsilon)$.

(4) *Observer model.* Because a probe is always initiated by an observer, the observability rate of beacon nodes is $r_b = 1$. A beacon node's positive and negative detection rates can be derived from the given parameters of the detector. Specifically, the negative detection rate is the possibility when a benign beacon nodes reports correct location information and its observer does not raise any alert. Liu and colleagues [2005] proposed the false positive rate P_{fp} as a parameter of the detector, which is the rate that a benign beacon node raises alerts against another benign beacon node. So we have the negative detection rate $r_n = 1 - P_{fp}$. The positive detection rate is the probability that the observer successfully detects it when a malicious beacon nodes provides false location data. This is the same as the detection rate P_r as defined by Liu and colleagues [2005].

Because each observer probes a beacon node independently, one potential risk is that the beacon node may behave different to different observers, so that the number of alerts from different observers will be significantly different statically. This will make the first assumption in Section 2.2 not valid anymore.

As mentioned above, the scheme of Liu and colleagues requires a detecting beacon node to use a completely new pseudoID for each probe. Therefore, a malicious beacon node cannot distinguish probes from different observers. Therefore, the number of alerts from uncompromised observers should still be statistically consistent during a period of time.

We intentionally omit the discussion of identification functions when modeling the above application with the proposed framework because their work cannot handle false alerts from compromised nodes. Though inconsistencies between alerts may be discovered, no solution is provided to reason which nodes are compromised. We will show how to design an efficient identification function based on the above framework in the next section. Before doing so, we use another example to further demonstrate how to map a real sensor network application to the proposed framework.

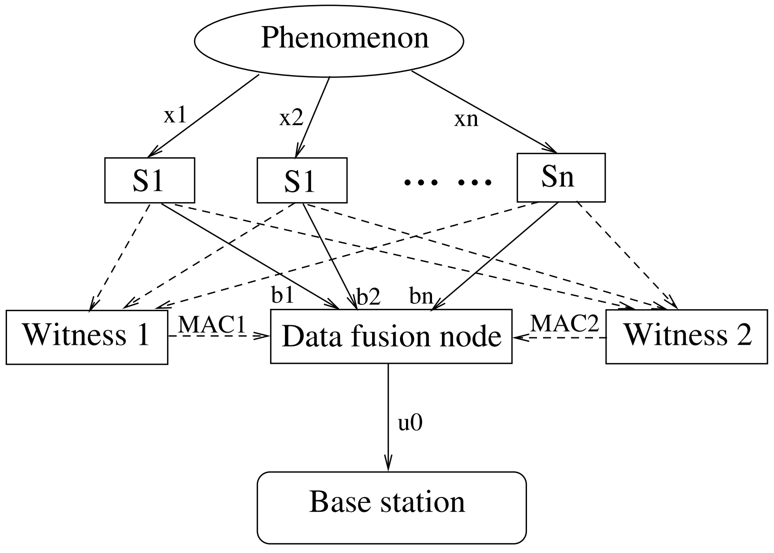
Example 2: Sensor Network Data Fusion. Data fusion is a common technique to reduce communication in sensor networks. Instead of having all the raw data sent to the base station, they are sent to nearby data fusion nodes, which aggregate them and forward the result further to the base station. Compromised data fusion nodes may send bogus aggregated information to the base station. Du and colleagues [2003a] proposed a witness-based mechanism to detect compromised data fusion nodes. The basic architecture of their approach is shown in Figure 3a. For a set of sensor nodes, there are one dedicated data fusion node and several witness nodes, all of which can receive sensing data from sensor nodes in the set. A witness node performs the same aggregation as the data fusion node, but only forwards the MAC of the result (using its shared key with the base station) to the data fusion node. The data fusion node then sends its result along with the witnesses' MACs to the base station. The base station checks the consistency between the data fusion node's result and the MACs of witnesses. This scheme assumes a shared secret key between each witness and the base station. So only the base station can check whether a MAC agrees with the data fusion result. Clearly, if a witness is compromised, it may also send bogus MACs.

This scheme can be modeled by our framework as follows.

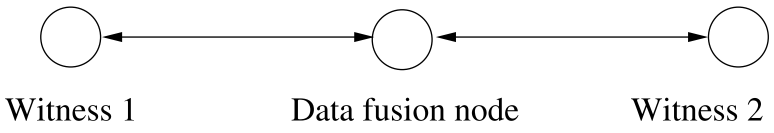
(1) Figure 3(b) shows the observability graph corresponding to Figure 3(a). If there is a disagreement between the data fusion node and a witness, it is equivalent that they raise alerts against each other. In the article by Du and colleagues [2003a], sensor nodes collecting raw data are assumed to be trusted. Therefore, the observability graph contains only data fusion nodes and witnesses. The observability graph will be more complicated if we consider the situation where a node may serve as a witness or an aggregator for more than one set of sensors.

(2) *Alerts.* If the MAC of a witness w_1 does not agree with the reported aggregation result of the data fusion node d_f for an event happened at time t, the base station can derive two alerts, (t, w_1, d_f) and (t, d_f, w_1).

(3) *Sensor behavior model.* We need to derive the reliability rate only for the data fusion node and witnesses. Suppose the communication channels between sensor nodes are reliable. Then the data fusion node and witness nodes will always receive the same data from the sensor node and get the same aggregation results. Thus, the reliability r_m of the data fusion node and the witness nodes is 1.



(a) The architecture of the detection mechanism for data fusion



(b) The observability graph of the detection mechanism for data fusion

Fig. 3. Observability graphs for the data fusion example.

(4) *Observer model.* If we assume the communication channels between sensor nodes are reliable, then the data fusion node and the witness nodes will always send data to the base station when an event happens. Thus the observability rate $r_b = 1$. The positive accuracy is the probability that the MAC sent by an uncompromised witness does not agree with the bogus aggregation result from a compromised data fusion node, which equals to $1-2^{-k}$, where k is the length of the MAC. The negative detection rate is the probability that the MAC from an uncompromised witness matches with the aggregation result from an uncompromised data fusion node, which equals to r_m.

Similar to the previous example, we omit the discussion of identification functions here, as their work cannot identify the compromised nodes either. In the following, we are going to present our approach to identifying compromised sensor nodes based on the above framework. Generally, we first derive the expected alert pattern from the sensor behavior model and observer model. Then, by comparing with the actual set of alerts, we identify those alerts that deviate from the expected behavior as abnormal. Finally, combining this information with the observability relationship, we design efficient identification functions such that the largest number of compromised nodes will be identified.

3. IDENTIFICATION OF COMPROMISED NODES

In this section, we present our approach to identifying compromised sensor nodes based on the above framework.

3.1 Overview of System Architecture

Let s_i be an observer of s_j. For each event at s_j, s_i will determine whether s_j has behaved correctly. If s_i believes s_j's behavior is suspicious, then an alert will be raised and sent back the base station. Here an event at s_j can be any behavior that the underlying detection system is interested in, according to different applications. For example, it can be a temperature value that s_j reports about the environment, or the fact that s_j has routed a packet for others, or the location information from s_j in response to the disguised beacon node s_i. Note that the functionalities of event monitoring, alert generation, and transmitting back to the based station are executed by the underlying detection mechanism, not by our model. Thus no additional communication and computation costs are imposed on the network.

In an ideal network, where the detection mechanism is completely accurate and the sensing and observing capabilities of sensor nodes are perfect, if s_i raises an alert against s_j, then at least one of them is compromised. Otherwise, if both behave normally, there should be no alerts between them. However, sensors are not assumed to have perfect sensing and monitoring capabilities. Therefore, the base station cannot draw any definite conclusion from a single alert. Instead, it needs to observe the alert pattern during a certain period of time to discover suspicious activities with high confidence. For this reason, the base station breaks the total operation time of the network into time intervals (each called a *time window*), and it reviews and reasons the alerts issued within each time window.

Within a given time window, the number of events of s_j is a random variable x, with a distribution $f_j(x)$. Given the sensor behavior model of s_j, observer model of s_i, and $f_j(x)$, we can derive the expected number of alerts raised by s_i against s_j in the time window when both of them are uncompromised. Then the base station can compare the number of alerts actually raised by s_i against s_j with the expected number within this window. Only when the former is higher than the latter with statistical significance should the base station consider it as abnormal.

For each pair of nodes that are considered abnormal, the base station will record an edge correspondingly in the observability graph (called an *abnormal edge*). As the base station has the knowledge of the global sensor topology, it can combine this information with all abnormal edges at hand to do further inference. Finally, we propose effective algorithms to reason the inferred graph and identify the compromised nodes.

Next, we will describe how the system works in details.

3.2 Alert Aggregation

Suppose s_j is a good node that functions normally according to the sensor behavior model, and s_i is a good observer to s_j; thus, its behavior follows the observer model. For every event at s_j, with probability r_m it will behave correctly. The observer s_i can detect s_j's behavior with probability r_b, and the probability that it issues a false alert stating that s_j behaves inappropriately is $1 - r_n$, according to the observer model. With probability $1 - r_m$, s_j will make mistakes, and s_i can detect this and issue a correct alert with probability r_p. These are the only two possibilities that s_i will raise an alert against s_j on a single event. Overall, the probability that s_i raises an alert against s_j on this single event is given by $C = r_b \cdot ((r_m \cdot (1 - r_n) + (1 - r_m) \cdot r_p)$. We interpret this as for each single event at s_j, there will be C alerts associated with it from s_i to s_j. In the sensor network localization, for example, each event is a probe from one beacon node to another beacon node. As shown in section 2.4, $r_b = 1$, $r_m = P(d < \epsilon)$, $r_n = 1 - P_{fp}$, and $r_p = P_r$. So $C = P(d \leqslant \epsilon) \cdot P_{fp} + (1 - P(d < \epsilon)) \cdot P_r$.

Let x denote the number of events at s_j within a time window T, which is a random variable, and $f_j(x)$ be the distribution of x. Then the distribution of alerts along the edge (s_i, s_j) (i.e., those raised by s_i against s_j) will be $f_{ij}(x) = f_j(\frac{x}{C})$. The expected number of alerts along (s_i, s_j) during the same period t will be $R_{ij}(t) = C \int^t f_{ij}(x)$.

During the time window T, if the number of alerts along the edge (s_i, s_j) is over $R_{ij}(T) + \delta$, 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. Here the parameter $\delta > 0$. Given the distribution of the expected number of alerts during T, δ can be computed as the $x\%$ confidence that the number of alerts is considered to be excessive. We leave the decision of $x\%$ to applications, as some applications may require a conservative reasoning, while others may not. The base station can even assign different $x\%$ to different observers according to their location, environment, functionality, and so on. We will not elaborate the details because they are highly application specific.

Our framework does not require the number of events of s_j monitored by each observer s_i to be the same. This will accommodate the underlying detector model to the largest extent. For example, in the beacon node application [Liu et al. 2005], different beacon nodes may probe a target beacon node different times within each time window. So the event distribution of s_j observable to s_i is different for each node s_i. But we can still derive the expected number of alerts along each edge (s_i, s_j) and tell if the edge is abnormal or not using the above reasoning.

Note that it is possible that s_j is not compromised but malfunctioning. But in this case we treat s_j as compromised anyway because information or services from s_j cannot be trusted anymore.

3.3 Graph Inference

Given a set of abnormal and normal edges, many existing trust functions in the literature can be applied to infer each sensor node's trustworthiness. However, because those functions are designed for general decentralized systems, they do not take into consideration the interaction topology between sensor

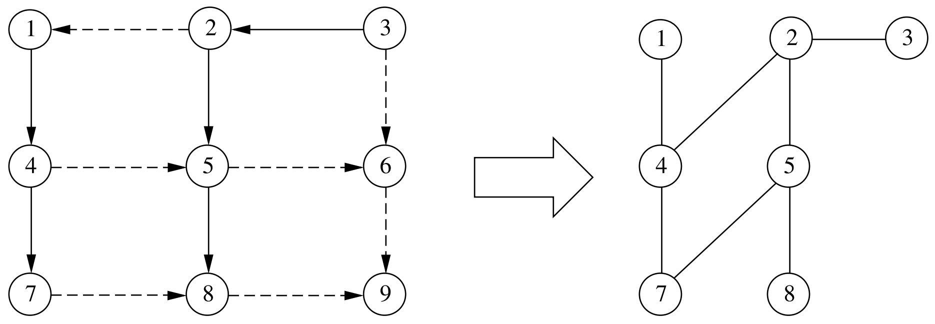


Fig. 4. An observability graph and its corresponding inferred graph.

nodes, which in fact provides valuable information to directly identify compromised sensor nodes. For example, suppose that there are no more than k compromised sensor nodes in a sensor network. If more than k abnormal edges involve a sensor node s, then we can know for sure that s is compromised. How to take advantage of such information to more effectively identify compromised sensor nodes is the focus of the rest of the paper.

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. Otherwise, the edge should be normal. 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 because s_l and s_i observe the activities of the same node s_j during the same period of time, according to our assumptions in Section 2.2. Information of such inconsistency is essential to identify compromised nodes.

Definition 3.1. Given an observability graph $G(V, E)$, let 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 $V' = \bigcup_{1 \leq i \leq k} \{s_i, s'_i\}$ and $E' = \{\{s_i, s'_i\} \mid 1 \leq i \leq k\}$.

Intuitively, if (s_i, s_j) are a suspicious pair, then at least one of them is compromised. Note that if a pair of nodes 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 4 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.

From the figure, one may wonder that, because $(2, 1)$ is normal and $(2, 5)$ is abnormal, shouldn't $\{1, 5\}$ be a suspicious pair too? The answer is no because 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 vice 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.

Definition 3.1 in fact offers two inference rules to identify pairs of sensor nodes such that at least one of them is compromised. A natural question is whether the two inference rules are complete. In other words, are there any other existing suspicious pairs that cannot be identified using the above two inference rules? The answer is no unless we impose further assumptions regarding the behavior of a compromised node.

A node s takes multiple responsibilities in a sensor network. Besides acting as a common sensor node, it also acts as observers of several other nodes $s_1, \ldots, s_k$. Without any assumptions on the application of the sensor network, a compromised node may behave arbitrarily when fulfilling different responsibilities. As the above example shows, a node may report false sensing data and meanwhile act normally as an observer, or vice versa. It may also issue bogus alerts to one node but not to others. Thus, the behavior of s in a time window can be represented as a vector $behave(s) = (sensing(s), obs(s, s_1), \ldots, obs(s, s_k))$. Each element in the vector can take the value of either g or b, corresponding to normal and abnormal behavior since respectively. A behavior of s is an assignment of values to the vector of s. Note that, because a compromised node may behave arbitrarily, there is no correlation between the value of each element. They can be assigned independently. Clearly, if at at least one of the elements in the vector must be assigned to b, then the node should be considered to be compromised. Otherwise, the node is considered as uncompromised, as long as there exist assignments that all its elements can be set to g.

Let $G(V, E)$ be an observability graph marked with normal and abnormal edges and $\mathcal{B} = \{behave(s_1), \ldots, behave(s_n)\}$ be a set of behaviors of all the nodes in G. We say $\mathcal{B}$ is consistent with G if the two conditions in Definition 3.1 are satisfied. That is, if there is an abnormal edge (s_1, s_2), then either $sensing(s_2) = b$ or $obs(s_1, s_2) = b$. Further, if (s_1, s) is normal and (s_2, s) is abnormal, then either $obs(s_1, s) = b$ or $obs(s_2, s) = b$.

THEOREM 3.2. *If two nodes s_1 and s_2 are not identified as a suspicious pair by Definition 3.1, then there exists a consistent set of behaviors of all the nodes where both s_1 and s_2 are uncompromised.*

PROOF. We first construct a consistent behavior set $\mathcal{B}$ as follows. For each node s, we set $sensing(s) = g$. For each abnormal edge (s', s), let $obs(s', s) = b$. For each normal edge (s'', s), let $obs(s'', s) = g$. It is easy to see that the resulting behavior set is consistent.

If in $\mathcal{B}$ both s_1 and s_2 are uncompromised, that is, all elements in $behave(s_1)$ and $behave(s_2)$ are assigned g, then we are done. Otherwise, s_1 must have an element $obs(s_1, s')$ that is assigned to b. Consider each node s' such that

$obs(s_1, s') = b$. Because s_1 and s_2 are not identified as a suspicious pair by definition 3.1, $s' \neq s_2$. If s_2 is also an observer of s', we must have $obs(s_2, s') = b$ as well. Therefore, we can set $sensing(s')$ to be b, and $obs(s_1, s')$ to be g. We also set $obs(s_2, s')$ to be g when s_2 is also an observer of s'. Next, for each edge (s'', s'), if it is abnormal, we set $obs(s'', s) = g$. Otherwise, set $obs(s'', s) = b$. By doing so, we remove one bad assignment in the behavior vector of s_1 and will not change any good assignment in the behavior vector of s_2 to bad. And the resulting behavior set is still consistent.

We keep doing the above modification for each node that s_1 and s_2 observe. The final behavior set is consistent, and both s_1 and s_2 are uncompromised. □

Note that a specific sensor network application may have further constraints on the behavior of a compromised node. For example, a node's observing behavior may be tied to its sensing behavior, that is, they have to be normal or abnormal at the same time. Without such application-specific properties, we may identify more suspicious pairs, which is outside of the scope of this paper, as we focus on a general application-independent framework for identifying compromised nodes.

Next, we discuss how to identify compromised nodes when taking the security estimation into consideration.

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 3.3. 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 satisfy all 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) $|S_b| \leq K$.

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 3.4. 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 $\bigcap_{1 \leq i \leq n} S_{bi}$ 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 3.5. 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).

If an identification function is optimal, then it identifies the largest number of compromised nodes without introducing any false positives. In other words, if a function F' returns any node s not in the compromised core, then we can always find a valid assignment such that s is not compromised in the assignment. This implies that F' may introduce false positives. 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, the compromised core may not achieve high detection rates because there may exist 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 trade off accuracy for eliminating more compromised nodes.

3.4 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 $s1$ issues a lot of alerts against $s2$, 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 3.6. *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 $|V_I|$.*

PROOF. When we assign the nodes into S_g, S_b, for each edge E_{ij}, at least one end point of them is in S_b. So S_b is essentially a vertex cover for I. $|S| \ge |V_I|$. □

We denote the size of the minimum vertex covers of an undirected graph G as C_G. Given a sensor node s, the neighbor 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 3.7. *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 $|\mathcal{N}_s| + C_{I'_s} > K$.*

PROOF. $\Rightarrow$: Suppose there exists some s that satisfies $|N_s| + C_{I'_s} > K$ and $s \in S_g$ for some assignment (S_g, S_b). Then we will have $N_s \subseteq S_b$. According to Lemma 3.6, we know the minimum number of malicious nodes in I'_s is $C_{I'_s}$. So we have $|S_b| \geq |N_s| + C_{I'_s} > K$. This contradicts with constraint 3 of Definition 3.3. So s must be in S_b under any assignment. Thus $s \in CompromisedCore(G, K)$.

$\Leftarrow$: If there exists $s \in CompromisedCore(G, K)$, that satisfies $|N_s| + C_{I'_s} \leq K$. Then we can always construct an assignment of S_b: $S_b = N_s \bigcup V_{I'_s}$. This assignment will satisfy all constraints of Definition 3.3, but it also introduces a contradiction: $s \in CompromisedCore(G, K)$, and $s \notin S_b$. So if $s \in CompromisedCore(G, K)$, it must satisfy $|N_s| + C_{I'_s} > K$. □

Intuitively, if we assume a sensor node s is uncompromised, then all its neighbors in I must be compromised. According to Lemma 3.6, 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 uncompromized, which means s is not in $CompromisedCore(I, K)$.

By Theorem 3.7, 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 because 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.

LEMMA 3.8. *Given an undirected graph G, $M_G \leq C_G \leq 2M_G$. And the bounds are tight [Vazirani 2001].*

COROLLARY 3.9. *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 [Micali and Vazirani 1980]. Algorithm 1 shows an efficient algorithm to approximate the compromised core. Because this algorithm does not assume any specific collusion model among compromised nodes, we call it the general identification algorithm.

Algorithm 1 AppCompromisedCore(I, K))

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

THEOREM 3.10. *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.*

PROOF. As described by Micali and Vazirani [1980], the complexity of finding the maximum matching of I'_s of any node s is $O(m\sqrt{n})$. Algorithm AppCompromisedCore will go through each node s in the system, so the total complexity is $O(mn\sqrt{n})$. □

3.5 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, for example, key revocation [Du et al. 2003b; Liu and Ning 2003]. However, there may still be suspicious pairs left that do not include any nodes in the compromised core. We call the graph composed of such pairs the residual graph.

We may trade off accuracy for eliminating more compromised nodes. Because 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 based on the information provided by the general framework. To reduce this worst-case false-positive rate, application-specific information or assumptions are needed.

The complexity of this phase is bounded by $O(m\sqrt{n})$ [Micali and Vazirani 1980], where m and n are the numbers of edges and vertices in an inferred graph respectively.

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.

3.6 Identification of Colluding Compromised Sensor Nodes

In general, collusion among compromised nodes will enable attackers to have a stronger influence on the sensor network and make it harder for the base station to identify them. The general identification algorithm does not assume any knowledge of the collusion among compromised nodes. Therefore, the worst case would be that the compromised nodes do collude, but the base station has to be conservative and use the general identification algorithm.

In this section, we consider the situation where the base station is aware of how compromised nodes will collude. Note that we are not advocating that this is a realistic situation, as in practice an attacker's strategy is most likely unknown. Instead, we intend to investigate how much the base station can do better with this extra knowledge, which will help us better understand the identification capability of the general algorithm. In other words, the identification capability of the base station with the extra knowledge of the attacker's strategy gives us an upper bound of that of the general identification algorithm.

For simplicity, in this section, we assume a strong collusion model where compromised nodes are coordinated with each other when monitoring events and raising alerts. In other words, compromised nodes do not raise alerts against each other, and alerts against a node from two compromised nodes are consistent. It is easy to see that in this case the inferred graph is bipartite, because an edge can be only between an uncompromised node and a compromised one. We call inferred graphs in this situation collusion-inferred graphs.

Let $G_1, \ldots, G_k$ be the connected components of a collusion inferred graph I. For each G_i, $1 \le i \le k$, let A_i, B_i denote the two sets of disjoint vertices in G_i such that vertices in one set are adjacent only to vertices in the other set. We call (A_i, B_i) the two-color partition of G_i. Without loss of generality, we assume $|A_i| \ge |B_i|$. Clearly, because I is collusion inferred, vertices in the same set are either all compromised or all uncompromised. Further, if vertices in one set are compromised (uncompromised), then vertices in the other set are uncompromised (compromised). Therefore, $\sum_{1\le i\le k} |B_i|$ becomes a lower bound of the number of compromised nodes in a network.

LEMMA 3.11. *Let $I(V, E)$ be a collusion-inferred graph and K be a security estimation. Given the two-color partition (A_i, B_i) of any connected component of I, $1 \le i \le k$, let Q_i be either A_i or B_i. We have $Q_i \subseteq$ UncompromisedCore(I, K) if and only if $|Q_i| + \sum_{j\ne i} \min(|A_j|, |B_j|) > K$.*

PROOF. $\Rightarrow$: For simplicity, we define the notation of sets in such a way that $|A_j| \ge |B_j|$ for all j as before. Suppose there exists $Q_i \subseteq$ *UncompromisedCore*(G, K), which satisfies $|Q_i| + \sum_{j\ne i} |B_j| \le K$. Then from Definition 3.4, $Q_i \in S_{gr}$ for all $1 \le r \le n$, where n is the number of valid assignments. Suppose the corresponding set in the same coloring partition with Q_i is P_i. Because P_i and Q_i are bipartite, we know that $P_i \in S_{br}$ for all $1 \le r \le n$, thus $P_i \subseteq$ *CompromisedCore*(G, K). Now we look at a particular assignment (S_{gx}, S_{bx}), $1 \le x \le n$, such that S_{bx} contains Q_i and all B_j, $1 \le j \le k$ except Q_i, where k is the number of two-color partitions and S_{gx} contains P_i

and all A_j, except Q_i. This assignment satisfies all constraints of Definition 3.3, but it also introduces contradiction: $Q_i \subseteq \mathit{UncompromisedCore}(G, K)$, thus $Q_i \subseteq S_{gj}$ for all $1 \leq j \leq n$, but $Q_i \subsetneq S_{gx}$ in this assignment. So if we have $Q_i \subseteq \mathit{UncompromisedCore}(G, K)$, it must satisfy $|Q_i| + \sum_{j \neq i} |B_j| > K$.

$\Leftarrow$: Suppose there exists some Q_i that satisfies $|Q_i| + \sum_{j \neq i} |B_j| > K$, and Q_i can be in some S_{by}, $1 \leq y \leq n$. Because each coloring partition will have a set in S_{by}, S_{by} will be composed of the following sets: $S_{by} = \{Q_i, A_r, \cdots, A_s, B_t, \cdots, B_m\}$. We thus have $|S_{by}| > |Q_i| + \sum_{j \neq i} |B_j| > K$, because $|A_i| \geq |B_i|$ for $1 \leq i \leq k$. This contradicts constraint 3 in Definition 3.3. So Q_i must be in S_{gy} for all $1 \leq y \leq n$. Thus $Q_i \subseteq \mathit{UncompromisedCore}(G, K)$. □

Intuitively, if Q_i is compromised, then $|Q_i| + \sum_{j \neq i} |B_i|$ gives a lower bound of the number of compromised nodes, which should be no more than K.

COROLLARY 3.12. *Given any connected component G_i of a collusion-inferred graph I and a security estimation K, $B_i \not\subseteq \mathit{UncompromisedCore}(I, K)$.*

PROOF. Suppose there exists some $B_i \subseteq \mathit{UncompromisedCore}(I, K)$, then we know $B_i \in S_{gi}$ for all assignment $1 \leq i \leq n$. Thus $A_i \in S_{bi}$ for all assignment $1 \leq i \leq n$, so we have $A_i \subseteq \mathit{CompromisedCore}(I, K)$. Now from Lemma 3.11, we have $|B_i| + \sum_{j \neq i} |B_j| > K$. Because $|A_i| \geq |B_i|$, we also have $A_i + \sum_{j \neq i} |B_j| > K$. Again from Lemma 3.11, A_i must be in $\mathit{UncompromisedCore}(I, K)$. This introduces contradiction. So $B_i \not\subseteq \mathit{UncompromisedCore}(I, K)$. □

THEOREM 3.13. *Given a collusion-inferred graph I with k connected components and a security estimation K, let (A_i, B_i), $1 \leq i \leq k$, be the two-color partition of each connected component. Let $\mathcal{S}_b = \{B_i \mid |A_i| + \sum_{j \neq i} |B_j| > K\}$, and $\mathcal{S}_a = \{A_i \mid |A_i| + \sum_{j \neq i} |B_j| > K\}$. Then $\mathit{CompromisedCore}(I, K) = \bigcup_{B_i \in \mathcal{S}_b} B_i$, and $\mathit{UncompromisedCore}(I, K) = \bigcup_{A_i \in \mathcal{S}_a} A_i$.*

PROOF. This is the direct observation based on Lemma 3.11 and Corollary 3.12. □

Given Theorem 3.13, it is straightforward to develop the algorithm that efficiently computes $\mathit{CompromisedCore}(I, K)$. The pseudocode is shown as Algorithm 2.

Algorithm 2 CollusionCompromisedCore(I, K)

```
//Input: $I$ is a collusion inferred graph
//       $K$ is a security estimation
//Output: the compromised core of $I$ with $K$
$S_b = \emptyset$
For each connected component $G_i$ of $I$
    Let $(a_i, b_i)$ be the two-color partition of $G_i$
    If $|a_i| + \sum_{j \neq i} |b_j| > K$
        $S_b = S_b \cup b_i$
Return $S_b$
```

THEOREM 3.14. *The complexity of algorithm* CollusionCompromisedCore *is* $O(n+m)$, *where n and m are the numbers of vertices and edges respectively in an collusion-inferred graph.*

PROOF. The complexity of finding all the two-color partitions from the collusion-inferred graph is $O(m)$, because we need to go through each edge to construct the partitions. The complexity of identifying the compromised core is $O(n)$, because the number of the connected components is in the order of $O(n)$, and we need to go through each component. Thus the total complexity of algorithm CollusionCompromisedCore is $O(n+m)$. □

After we have identified the compromised core for the bipartite graph, we achieve the goal of identifying the largest number of compromised nodes without introducing any false positives. For the residual graph, we can use the same maximum matching algorithm as we proposed in Section 3.5 to trade off false positives for detection rates.

4. EXPERIMENTS

In this section, we design a set of experiments to evaluate the effectiveness of our algorithms.

4.1 Experiment Methodology

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°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. We assume that, once a network is deployed, sensors' location information can be collected through localization techniques. Therefore, the base station is able to construct an observability graph accordingly.

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 [Crossbow Technology Inc. 2003]. The overall operation time is 24 hours, but the base station will review the system every hour. So within each time window of one hour, there will be 60 data reports from each sensor node. In the simulation, we do not consider such effects as the message loss, as it has already been modelled by the parameters as r_b, r_m, and r_n.

Unless otherwise stated, we assume $10\% \sim 15\%$ of the nodes in the network are compromised. The security estimation is $K = 15\%N$, 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. In this section we show the detection rates and false-positive rates separately in different figures. We do not plot the receiver operating characteristic curve (ROC) of our algorithm because, given an observability graph and a set of alerts, there is no other parameters in our approach to adjust the identification results.

The effectiveness of our algorithm will be affected by the actual deployment of the sensor nodes, especially the observability graph. Thus, for each simulation, we generate ten random deployments of sensor networks, following the parameters specified for each one's setup. The detection rates and false-positive rates are averaged over the results on these ten random networks. We have identified through simulations that the 95% confidence interval of the detection rate and false-positive rates are within 1% across the ten random networks, so we believe ten should be enough.

We first act conservatively and assume that compromised nodes in fact follow the strong collusion model, but the base station does not know this and cannot use 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. As a comparison, we will also see how well we can do if we know the fact that the attackers are colluding and evaluate the *CollusionCompromisedCore* algorithm followed by the maximum matching approach. We call it *bipartite+mm* in the experiment.

We further compare our approach with the simple voting mechanism. Among a node *s*'s neighbors, if those that raise alerts against *s* are more than those that do not, then *s* is considered as compromised.

We also compare our approach with EigenTrust [Kamvar et al. 2003] and PeerTrust [Xiong and Liu 2002], two well-known reputation-based trust functions for P2P systems. Though there are many trust functions proposed for P2P networks and semantic webs, many of them are decentralized and subjective, in the sense that an entity's trust value varies depending on who is the trust evaluator [Golbeck and Hendler 2004; Lee et al. 2003; Richardson et al. 2003; Yu and Singh 2002]. They are not suitable for centralized identification of compromised nodes in sensor networks. 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 and bipartite+mm.

The idea of EigenTrust is similar to the PageRank algorithm [Lawrence et al. 1998]. In EigenTrust, the number of satisfactory and unsatisfactory transactions between each pair of entities is collected to construct a matrix. One special property of the matrix is that the entries in each row add up to 1.

The matrix is repetitively multiplied with an initial vector until it converges. The initial vector is a predefined parameter that corresponds to the default trust value of each entity. Each entry in the converged vector represents an entity's final trust value. Because the trust values of all nodes always add up to 1, $1/N$ is the average trust value of sensor nodes in the network. In the experiment, we identify the K nodes with the lowest trust values as compromised, unless their trust values are over $1/N$.

In PeerTrust, an entity's trust value is the normalized number of satisfactory services over the total number of transactions in which the entity takes part. It is further weighted by the trust values of the transaction feedback issuers and the quantity of each transaction. Similar to EigenTrust, the global trust values for all entities are also obtained through iterative matrix multiplication until they converge. In general, PeerTrust can be viewed as a majority voting scheme weighted by voters' trust values.

Traditional fault diagnosis PMC models are not applicable to sensor networks, as we have analyzed in Section 1, and we do not compare them.

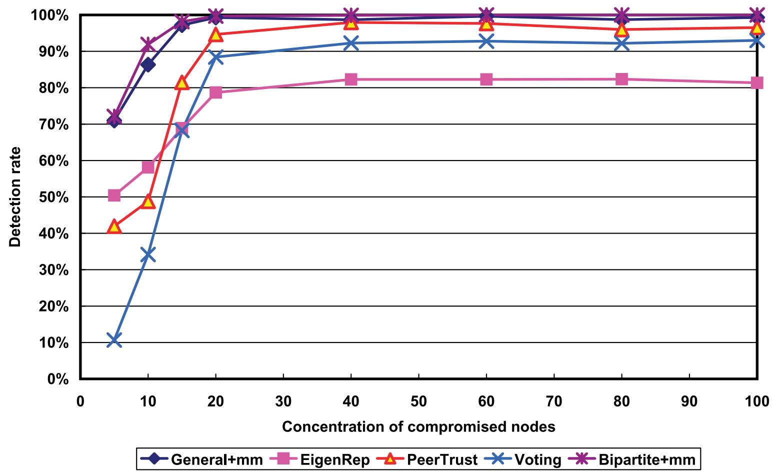
We conduct four sets of experiments to evaluate the impacts of the following factors on the effectiveness of the above approaches.

4.2 Local Majority

Figure 5 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 involve only 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, bipartite+mm, and PeerTrust. The reason is that EigenTrust relies on the existence of pretrusted peers to identify malicious collectives, which correspond to colluding compromised nodes in our setting. Without pretrusted 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



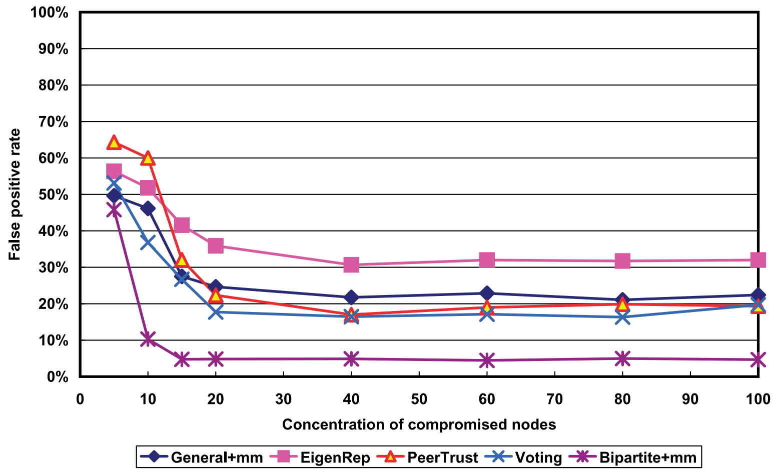


Fig. 5. The impact of the concentration of compromised sensor nodes.

of a compromised node, on the average no more than one 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 node 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 compromise 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.

Last, bipartite+mm always has the highest detection rate and lowest false-positive rate. This shows that, when we have the exact information of the attacker's behavior, the dedicated identification algorithm can be very effective.

4.3 Sensor Node Density

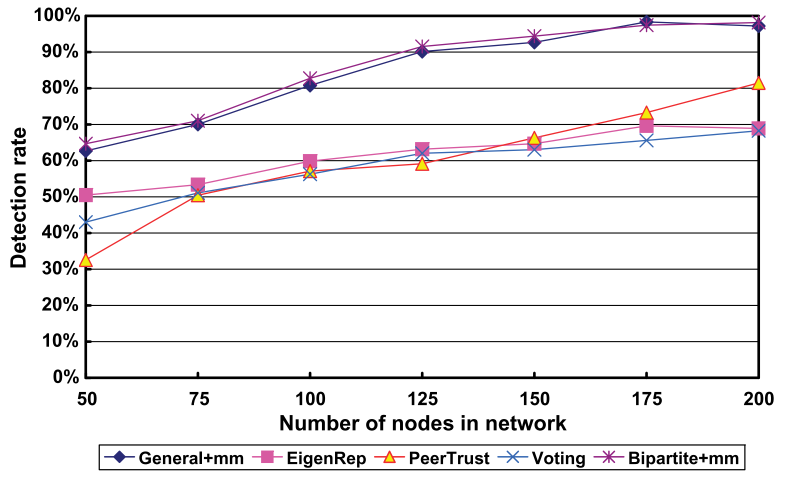
In the next experiment, we vary the number of sensor nodes in the area from 50 to 200. As we have seen from previous experiment, when there is no collusion among compromised nodes, all algorithms have high detection and false-positive rates, and our algorithm outperforms the other three algorithms when there exists a strong collusion. Thus we report the experimental results, which set the concentration of compromised nodes to be 15, not too strong a case. We have also tried other concentration parameters and observed similar trends. Figures 6a and 6b 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 lower 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 to be detected. The second observation is that general+mm and bipartite+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. A further study tell us that on average, when there are 100 nodes in this area, 16.6 nodes have only one observer. When there are 150 nodes in the area, 10 of them have only one observer. When we deploy 200 nodes in the area randomly, only 5.2 nodes on average will have one observer.

The third observation, is that we see that general+mm detects much more compromised nodes with similar false-positives than all other approaches, because it takes the unique properties of sensor networks into consideration so that it is more resilient to compromised nodes forming local majorities. Finally, the bipartite+mm algorithm is still the best, as it has additional information about attacker's behavior.

4.4 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



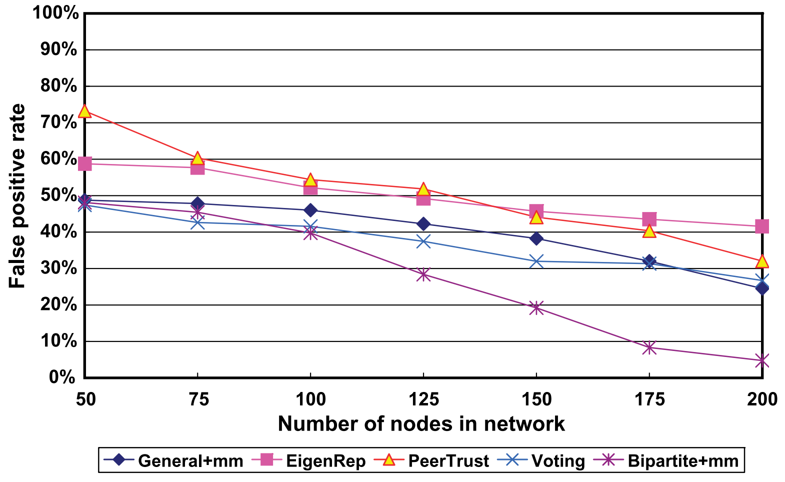


Fig. 6. The impact of the deployment density of sensor nodes.

available. The next experiment evaluates how the accuracy of a security estimation affects the effectiveness of different approaches. The total number of sensor nodes is 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. The 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 7.

We see that general+mm still achieves 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

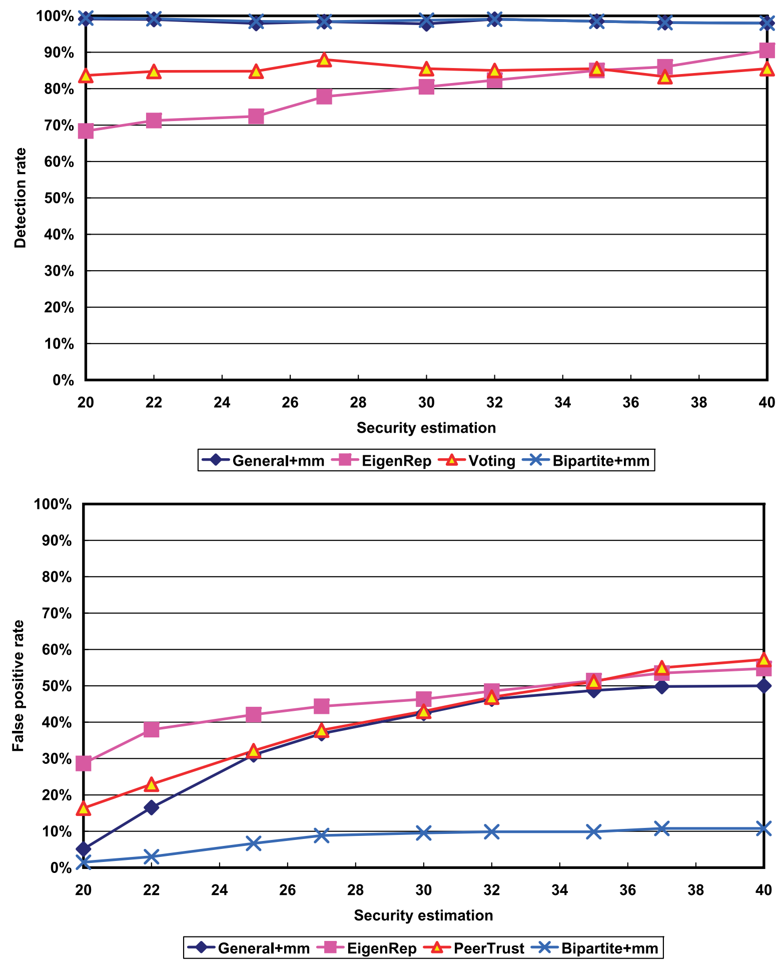


Fig. 7. The impact of the accuracy of security estimation.

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. For bipartite+mm, even when K is set as twice the real number of compromised nodes, the false-positive rate is still not high. This is because, when we reason on the set of alerts and build the inferred graph, we take into consideration not only the nodes that are directly involved in the alerts but also some other nodes that are not involved in any alerts but can observe those involved nodes. When having more nodes, chances are there will be more nodes in each partition of the bipartite graph; thus, the judgment condition $|a_i| + \sum_{j \neq i} |b_j| > K$ in algorithm CollusionCompromisedCore is more easily satisfied than the condition $n_s + m > K$ in algorithm AppCompromisedCore.

In summary, our experiments show that the proposed two-phase approach in most cases achieves high detection rates with low false-positives, even when sensor nodes are relatively loosely deployed, and compromised nodes form strong local majorities. Also, because they are designed to accommodate the uniqueness of sensor networks, they consistently outperform EigenTrust and PeerTrust, two well-known reputation-based trust management schemes for general decentralized systems, as well as the simple voting mechanism. This is especially true when compromised nodes form strong local majorities.

Our experiments also indicates how alerts can be implicitly inferred from application data for some specific applications, as in our example, thus do not introduce any additional communication or computation cost at the network.

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. As in 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 that 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 Lemma 3.6. 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.

4.5 Network Evolution

After some compromised nodes are identified and removed, there will be fewer nodes remaining in the system, which will inevitably affect the effectiveness of our scheme. In this section, we conduct experiments to evaluate the degradation of proposed scheme when an attacker continues to compromise more sensor nodes. Specifically, we set the number of sensor nodes in the area to be 200 initially. In each time window, the attacker compromises 15 more nodes. These nodes are chosen from a random new center (x, y) within the area, with concentration of 15. We set the security estimation $K = 20$ to reflect our knowledge that no more than 20 nodes will be compromised within each time window. When the total number of identified compromised nodes is bigger than K, an exception will be thrown to indicate that we need to reevaluate our security estimation. We run both general+mm and bipartite+mm. Figure 8 shows the detection rate and false-positive rate at the end of each time window when the base station uses the two algorithms to identify compromised nodes.

We see that, as time goes on, the detection rate of both general+mm and bipartite+mm decrease, and the false positive rates increase. This is expected as the density of the network gets smaller when more and more nodes are

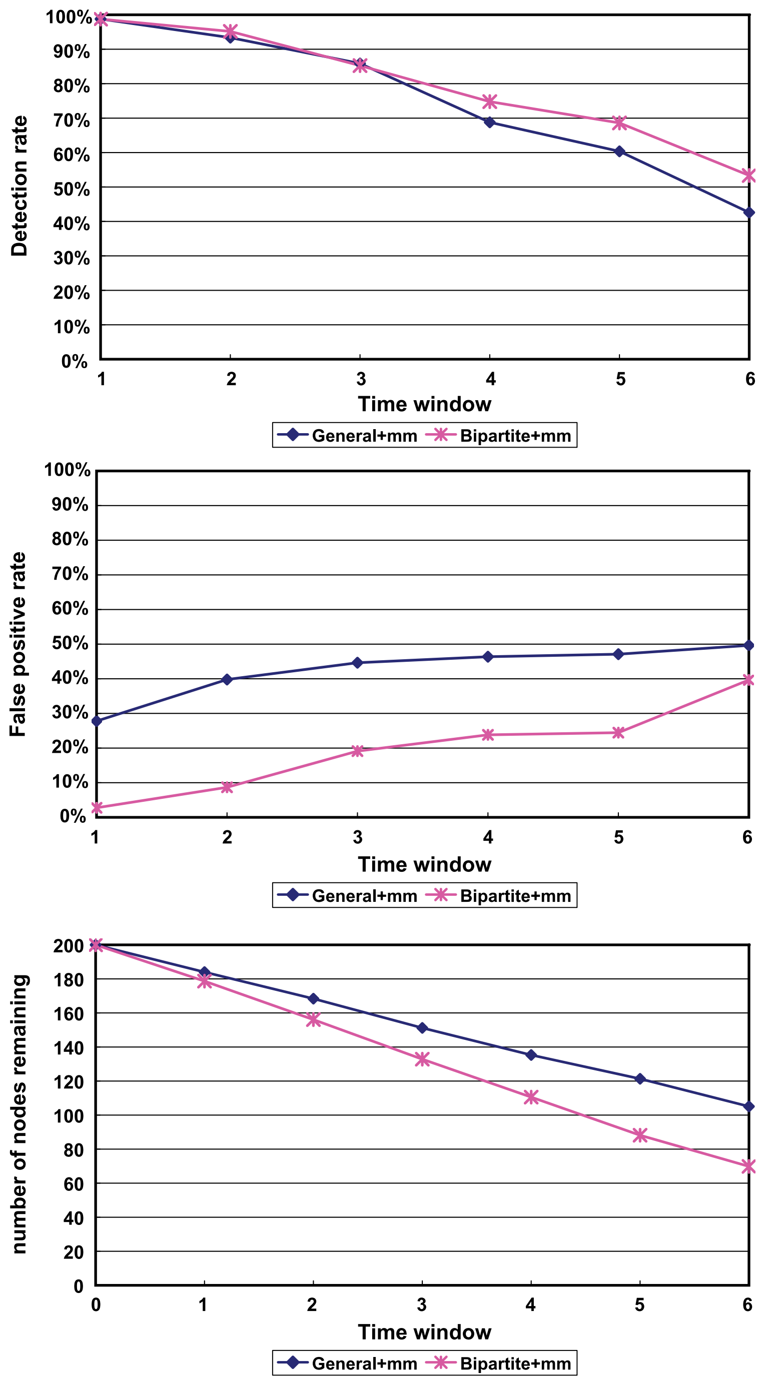


Fig. 8. The effectiveness of the proposed scheme when an attacker continues to compromise sensor nodes.

excluded from the system. To illustrate this, the number of nodes remaining in the network after we exclude those identified as compromised is shown in Figure 8c. As we have shown in Section 4.3, when the density of a network decreases, a node will have fewer observers, which makes it harder to definitely identify a compromised node.

The result of the experiment suggests the importance of network reconfiguration. Once some compromised nodes are removed, it is necessary to deploy new sensor nodes into the network so that each node is observed by a sufficient number of other nodes. Otherwise, an attacker can selectively compromise those nodes that are insufficiently observed, and avoid being identified. We briefly discuss network reconfiguration in Section 5.

5. DISCUSSION

5.1 Attacker's Strategy

The identification algorithm is not secret. Thus it is important to investigate that, with the knowledge of the identification algorithm, what an attacker should do to compromise nodes and maximize its influence in the sensor network. The measure of the influence of compromised nodes often depends on the functionality of specific applications. In this article, we adopt a simple definition based on the following observation. After some nodes are identified as compromised, they will be removed from the network. The data collected by them will not be used in a sensor network application. Thus, after the identification algorithm is used, the bigger the portion of sensor nodes that are controlled by an attacker, the larger influence the attacker has.

Specifically, a compromised node may have the following malicious activities: sending false data, sending false alerts against an uncompromised node, and not sending alerts when it should according to a sensor network's detection mechanism. A compromised node may take multiple malicious actions during a period of time. The goal of the attacker is to maximize false information introduced into the systems. In other words, the more nodes sending false data that are not detected, the larger the influence the attacker has in the sensor network.

If a compromised node s_1 sends false data and one of its observers s_2 is not compromised, then s_2 will issue alerts against s_1 to the base station, and $\{s_1, s_2\}$ forms a suspicious pair in the inferred graph. Due to the maximum matching in the second phase of the identification algorithm, s_1 and s_2 may both be identified as compromised. Thus, to be sure that s_1 is definitely not identified as compromised by our identification algorithm, the attacker also needs to compromise all the observers of s_1. In some sense, the compromised observers of s_1 serve as the protectors of s_1 so that its false data can be accepted by the base station. The protectors may still send legitimate data to the base station. They just do not issue alerts against s_1. By doing so, neither s_1 nor its observers will be included in any suspicious pairs. Therefore, s_1 will definitely be treated as uncompromised by the identification algorithm.

Generally, we have the following definition.

Definition 5.1. Let G be an observability graph and S be a set of vertices in G. We say $S' \subseteq S$ is a protected set of S if, for any $s \in S'$, all the observers of s belong to S. The protection capability of S is the size of the maximum protected set of S.

Intuitively, the protection capability of S corresponds to the maximum amount of false data that an attacker can definitely inject to the base station once S is compromised.

Given an observability graph G, the attacker may face the following two types of problems. First, suppose the attacker has the resources to compromise only n nodes. Then how can one determine a set of n nodes in G so that they have the maximum protection capability? Second, suppose the attacker would like to inject false data from at least t nodes without being detected; then what is the minimum number of sensor nodes it has to compromise?

THEOREM 5.2. *The problem to maximize an attacker's protection capability after compromising n nodes is NP complete.*

PROOF. First let us look at the graph-cutting problem [Marx 2004]: Given a graph $G(V, E)$ and two integers k and l, is there a partition $V = X \cup S \cup Y$ such that $|X| = l$, $|S| \leq k$ and there is no edge between X and Y? This problem is equivalent to the graph-cutting problem where $l = |V| - n$. It has been shown by Marx that graph-cutting problem is NP complete by reducing the minimum vertex covering problem to this problem [Marx 2004]. □

THEOREM 5.3. *The problem to minimize the number of compromised nodes to achieve protection capability t is NP complete.*

PROOF. Similar to the proof above, this problem is equivalent to the graph-cutting problem where $l = t$. □

The above results are bad news for attackers, fortunately. With limited resources, it is very hard for attackers to inject maximum false information. Also, to inject a certain amount of false information, the attacker may have to end up compromising more nodes than necessary, which requires more effort. On the other hand, it is not clear yet whether there exists a good approximation algorithm to the above problems. This is one of the topics that we will investigate in our future work.

5.2 Attacks

The proposed framework and algorithms focus on identifying compromised nodes through reasoning about alerts between sensor nodes. One possible attack is that an attacker may repetitively trigger events that can only be monitored by a node s. Thus, the data reported by s may be significantly different from that of others. This may cause many alerts against s, and have s identified as compromised. The essential reason for such attack is that the detection mechanism cannot tell the difference in the information of a real phenomenon from pure bogus information generated by a compromised node. This problem cannot be handled by the general framework. Instead, it requires more accurate application detection mechanisms, for example, having sensor nodes more densely deployed so that any event can be monitored by multiple nodes.

5.3 Network Reconfiguration

Once compromised nodes are identified, this calls for mechanisms to effectively mitigate their impacts. One straightforward method is to remove those sensor nodes from the network through, for example, key revocation. In some situations however, this may not be enough. Sensors may provide several services in a single application, such as routing, data sensing, and data aggregation. Removing sensors from a network also removes services from them, which may have a significant impact on the functionality of an application. Moreover, as shown by the experiment in Section 4.5, when there are fewer nodes in the area, it is also more vulnerable to attacks. Therefore, it is often necessary to reconfigure the service-providing relation between existing sensors or to deploy new sensors. Sensor network reconfiguration has several goals, including preserving the functionality of a sensor network, minimizing reconfiguration costs, and improving the overall trustworthiness of a sensor network. A suitable reconfiguration cost model for sensor networks is essential to achieve the above goals.

5.4 Decentralized Approaches

In this article, we adopt a centralized approach, that is, the base station collects alerts and identifies potentially compromised nodes. A centralized approach usually offers better accuracy in identifying compromised and malfunctioned nodes, because it has a global view of the network. A decentralized approach (as described by Ganeriwal and Srivastava [2004]) is a possible alternative, which limits alerts to be exchanged between nearby nodes. A decentralized approach may incur lower communication costs. But without global information, it is in general more difficult to deal with the local majority and collusion of compromised nodes. How to design a lightweight decentralized approach to accurately identify compromised nodes instead of just tolerating them is a challenging problem.

6. RELATED WORK

Much work has been done to provide security primitives for wireless sensor networks, including practical key management [Chan et al. 2003; Du et al. 2003b; Eschenauer and Gligor 2002; Liu and Ning 2003], broadcast authentication [Liu et al. 2003; Perrig et al. 2000, 2001], and data authentication [Hu and Evans 2003; Przydatek et al. 2003; Zhu et al. 2004] as well as secure in-network processing [Deng et al. 2003]. The work of this article 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 [Du et al. 2003a; Hu and Evans 2003; Przydatek et al. 2003] through, for example, sampling and redundancy. But they do not provide mechanisms to accurately identify compromised sensor nodes, which is the focus of this article.

Reputation-based trust management has been studied in different application contexts, including P2P systems, wireless ad hoc networks, social networks, and the Semantic Web [Aberer and Despotovic 2001; Kamvar et al. 2003; Lee et al. 2003; Mui et al. 2002; Richardson et al. 2003]. Many trust inference schemes have been proposed. They differ greatly in inference methodologies, complexity, and accuracy. As discussed earlier, 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 and Srivastava [2004] propose to detect abnormal routers in sensor networks through reputation mechanism. It is assumed that a sensor's routing quality can be observed by nearby sensors through a watchdog mechanism. Ganeriwal and Srivastava adopt a decentralized trust inference approach. Sensors evaluate each other's trustworthiness by acquiring feedback information from nearby sensors. Ganeriwal and Srivastava's work 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. Finally, their work is application specific and cannot be easily applied to other sensor network applications.

The problem of detecting faulty nodes in multiprocessor systems has been studied for a long time. The pioneering work is the PMC model proposed by Preparata and colleagues [1967]. Efficient diagnosing algorithms to identify faulty nodes have also been proposed [Araki and Shibata 2003; Dahbura and Masson 1984; Fuhrman 1996; Sullivan 1988]. However, all these algorithms assume that the test assignments follow some special topologies to ensure the system is t-diagnosable in the first place, which means all faulty nodes in the system can be identified as long as there are at most t faulty nodes within it. These algorithms cannot be applied to sensor networks due to their topology requirements, as deployments in sensor networks are often ad hoc.

Some variants of the PMC relaxed from permanent faults to intermittent faults [Dahbura et al. 1987; Kozlowski and Krawczyk 1991], but they need certain assumptions, that is, the faulty nodes will be faulty following certain probabilities [Dahbura et al. 1987], or the number of incorrect outcomes is bounded [Kozlowski and Krawczyk 1991]. In the most general cases, it is shown by Dahbura and Masson that the problem is NP complete [Dahbura and Masson 1983a; 1983b].

Previous work on Byzantine fault detection generally focuses on the designing of communication protocols or message/detector structures, so that a proper voting mechanism can lead to the exposure of Byzantine generals [Ho et al. 2004; Lamport et al. 1982]. Sensor nodes are often randomly deployed; thus, solutions in this area are not applicable in sensor networks either.

7. CONCLUSION

Wireless sensor networks are often deployed unattended in open environments. Sensors are subject to capture by attackers and thus more likely to be compromised. Once keying materials are recovered, an attacker may be able to impersonate compromised nodes completely and inject false information into sensor network to influence the outcome of an application.

In this article, 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. Built on the alert-based detection mechanisms provided by applications, our framework does not introduce additional communication and computation costs to the network. 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 is 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. Our algorithm maintains good performances even if we do not have a good estimation of the secrecy of the system.

We plan to extend this work in the following directions. First, we are interested in designing a cost model for sensor network reconfiguration to mitigate the effect of compromised nodes. The model should include possible reconfiguration mechanisms and consider the multiple functionalities provided by a sensor network and their dependency. Second, we plan to investigate lightweight decentralized approaches and systematically analyze its benefits and inherent weakness when compared with centralized approaches. Third, we also plan to explore the Bayesian model reasoning and assign a probability for each edge to infer its likelihood to be abnormal, rather than the current binary model.

REFERENCES

ABERER, K. AND DESPOTOVIC, Z. 2001. Managing trust in a peer-2-peer information system. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*.

ARAKI, T. AND SHIBATA, Y. 2003. (t, k)-diagnosable system: A generalization of the pmc models. *IEEE Trans. Comput. 52,* 7.

BOSE, P., MORIN, P., STOJMENOVIC, I., AND URRUTIA, J. 2001. Routing with guaranteed delivery in ad hoc wireless networks. *ACM Wirel. Netw. 7,* 6, 609–616.

CAMTEPE, S. AND YENER, B. 2004. Combinatorial design of key distribution mechanisms for wireless sensor networks. In *9th European Symposium On Research in Computer Security (ESORICS'04)*.

CHAN, H., PERRIG, A., AND SONG, D. 2003. Random key predistribution schemes for sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy(SP'03)*.

CROSSBOW TECHNOLOGY INC. 2003. *MTS/MDA Sensor and Data Acquisition Boards User Manual*.

DAHBURA, A. AND MASSON, G. 1983a. Greedy diagnosis of an intermittent-fault/transient-upset tolerant system design. *IEEE Trans. Comput. C-32,* 10, 953–957.

DAHBURA, A. AND MASSON, G. 1983b. Greedy diagnosis of hybrid fault situations. *IEEE Trans. Comput. C-32,* 8, 777–782.

DAHBURA, A. AND MASSON, G. 1984. An $o(n^{2.5})$ fault identification algorithm for diagnosable systems. *IEEE Trans. Comput. C-33,* 6, 486–492.

DAHBURA, A., SABNANI, K., AND KING, L. 1987. The comparison approach to multiprocessor fault diagnosis. *IEEE Trans. Comput. C-36,* 3, 373–378.

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

DENG, J., HAN, R., AND MISHRA, S. 2004. A robust and light-weight routing mechanism for wireless sensor networks. In *Proceedings of the Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS)*.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. K. 2003a. A witness-based approach for data fusion assurance in wireless sensor networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. K. 2003b. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*.

DU, W., FANG, L., AND NING, P. 2005. Lad: Localization anomaly detection for wireless sensor networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*.

ESCHENAUER, L. AND GLIGOR, V. D. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*.

FUHRMAN, C. P. 1996. Comparison-based diagnosis in faulttolerant, multiprocessor systems. Ph.D. thesis, Department of Computer Science, Swiss Federal Institute of Technology in Lausanne (EPFL).

GANERIWAL, S. AND SRIVASTAVA, M. B. 2004. Reputation-based framework for high integrity sensor networks. In *Proceedings of the ACM Security for Ad-Hoc and Sensor Networks (SASN'04)*.

GOLBECK, J. AND HENDLER, J. 2004. Accuracy of metrics for inferring trust and reputation in semantic Web-based social networks. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*. Northamptonshire, U.K.

HO, T., LEONG, B., KOETTER, R., MEDARD, M., EFFROS, M., AND KARGER, D. 2004. Byzantine modification detection in multicast networks using randomized network coding. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*.

HU, L. AND EVANS, D. 2003. Secure aggregation for wireless networks. In *Proceedings of the Workshop on Security and Assurance in Ad Hoc Networks*.

KAMVAR, S., SCHLOSSER, M., AND GARCIA-MOLINA, H. 2003. EigenRep: Reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference*.

KOZLOWSKI, W. AND KRAWCZYK, H. 1991. A comparison-based approach to multicomputer system diagnosis in hybrid fault situations. *IEEE Trans. Comput. C-40,* 11, 1283–1287.

LAMPORT, L., SHOSTAK, R., AND PEASE, M. 1982. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4,* 3.

LAWRENCE, R., SERGEY, B., RAJEEV, M., AND TERRY, W. 1998. The PageRank citation ranking: Bringing order to the Web. Tech. rep., Department of Computer Science, Stanford University.

LEE, S., SHERWOOD, R., AND BHATTACHARJEE, B. 2003. Cooperative peer groups in NICE. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*.

LIU, D. AND NING, P. 2003. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*.

LIU, D., NING, P., AND DU, W. 2003. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03)*.

LIU, D., NING, P., AND DU, W. 2005. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS'05)*.

LIU, D., NING, P., AND LI, R. 2005. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inform. Syst. Secur. 8*, 1.

MARX, D. 2004. Parameteried complexity of constraint satisfaction problems. In *Proceedings of the 19th Annual IEEE Conference on Computational Complexity*.

MICALI, S. AND VAZIRANI, V. 1980. An $o\sqrt{|V|}|e|$ algorithm for finding maximum matchings in general graphs. In *Proceedings of the 21st Symp. Foundations of Computing*.

MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. 2002. A computational model of trust and reputation. In *Proceedings of the 35th Hawaii International Conference on System Science*.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2000. Effient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Security and Privacy*.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. D. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual ACM International Conference on Mobile Computing and Networks (MobiCom'01)*.

PREPARATA, F. P., METZE, G., AND CHIEN, R. T. 1967. On the connection assignment problem of diagosable systems. *IEEE Trans. Electron. Comput. 16,* 6, 848–854.

PRZYDATEK, B., SONG, D., AND PERRIG, A. 2003. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*.

RICHARDSON, M., AGRAWAL, R., AND DOMINGOS, P. 2003. Trust management for the Semantic Web. In *Proceedings of the 2nd International Semantic Web Conference*.

SULLIVAN, G. F. 1988. An $o(t^3 + |e|)$ fault identification algorithm for diagnosable systems. *IEEE Trans. Comput. 37,* 4.

VAZIRANI, V. V., Ed. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.

XIONG, L. AND LIU, L. 2002. Building trust in decentralized peer-to-peer electronic communities. In *Proceedings of the 5th International Conference on Electronic Commerce Research (ICECR)*.

YE, F., LUO, H., LU, S., AND ZHANG, L. 2004. Statistical en-route filtering of injected false data in sensor networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*.

YU, B. AND SINGH, M. P. 2002. An evidential model of distributed reputation management. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

ZHANG, Q., YU, T., AND NING, P. 2006. A framework for identifying compromised nodes in sensor networks. In *Proceedings of the 2nd IEEE Communications Society/CreateNet International Conference on Security and Privacy in Communication Networks (SecureComm'06)*.

ZHU, S., SETIA, S., JAJODIA, S., AND NING, P. 2004. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Proceedings of the IEEE Symposium on Security and Privacy,* 260–272.

Received February 2007; revised August 2007; accepted September 2007