Attack-Resilient Time Synchronization for Wireless Sensor Networks

Hui Song, Sencun Zhu, and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {hsong,szhu,gcao}@cse.psu.edu

***Abstract*— The existing time synchronization schemes in sensor networks were not designed with security in mind, thus leaving them vulnerable to security attacks. In this paper, we first identify various attacks that are effective to several representative time synchronization schemes, and then focus on a specific type of attack called *delay attack*, which cannot be addressed by cryptographic techniques. Then, we propose two approaches to detect and accommodate the delay attacks. Our first approach uses the generalized extreme studentized deviate (GESD) algorithm to detect multiple outliers introduced by the compromised nodes; our second approach uses a threshold derived using a time transformation technique to filter out the outliers. Finally, we show the effectiveness of these two schemes through extensive simulations.**

I. Introduction

Many sensor network applications require time to be synchronized within the network. Examples of such applications include mobile object tracking [1], [2], data aggregation, TDMA radio scheduling, message ordering, multicast source authentication protocol [3], to name a few. Consider the application of mobile object tracking, in which a sensor network is deployed in an area of interest to monitor passing objects. When an object appears, the detecting nodes record the detecting location and the detecting time. Later, these location and time information are sent to the aggregation node which estimates the moving trajectory of the object. Without an accurate time synchronization, the estimated trajectory of the tracked object could differ greatly from the actual one. Similarly, we can see the importance of time synchronization for the operations of other sensor network applications.

All network time synchronization methods rely on some sort of message exchanges between nodes. Nondeterminism in the network dynamics such as physical channel access time or operation system overhead (e.g., system calls), makes the synchronization task challenging in sensor networks. In the literature, many schemes have been proposed to address the time synchronization problem [4], [5], [6], [7]. These schemes involve the exchange of multiple time synchronization messages among multiple sensor nodes [4] or between two sensor nodes [5] to be synchronized. However, none of them was designed with security in mind, even though security has been identified as a major challenge for sensor networks [8]. Actually, even if an adversary is capable of destroying some or all sensor nodes, it may opt for other more severe attacks, since it is more dangerous to take actions based on some false sensor data than without any data. For example, if an adversary can attack the time synchronization protocol so that the estimated direction of a mobile object is contrary to its actual direction, a wrong or even risky action may be taken and many system resources may be wasted. Thus, when a sensor network is deployed in an adversarial environment such as a battlefield, the time synchronization protocol is an attractive target to the adversaries.

In this paper, we first identify several security attacks an adversary can launch against a non-secure time synchronization protocol. For instance, an attacker can replay old synchronization messages, drop, modify, or even forge exchanged timing messages. Since many of these attacks can be addressed by employing appropriate cryptographic techniques, we focus on a specific type of attack called *delay attack* which cannot be addressed by the cryptographic techniques. In the delay attack, a malicious attacker (or a compromised node) deliberately delays the transmission of time synchronization messages to magnify the offset between the time of a malicious node and the actual time. All the current time synchronization schemes [4], [5], [6], [7] are vulnerable to this attack in one way or another.

We propose two approaches to detect and accommodate the delay attacks. Our first approach uses the generalized extreme studentized deviate (GESD) algorithm to detect the outliers introduced by malicious nodes. If there is no malicious node, the time offsets among the sensor nodes should follow the same (or similar) distribution or pattern. For their attacks to be effective, malicious nodes typically report their time offsets much larger than those from the benign nodes, leaving their reported values suspicious. Our second approach uses a time transformation technique, which enables every node to derive an upper bound of the time offset that is acceptable to it, thereby filtering out the outliers. We discuss the merits as well as the limitations of each approach, and evaluate the effectiveness of these two schemes through extensive simulations.

The rest of the paper is organized as follows. The next section describes the related work and discusses various attacks which are addressable using cryptographic techniques. In Section III, we identify and discuss a new attack called *delay attack*. Section IV presents the system model and assumptions. In Section V, we present the outlier-based approach. Section VI presents the threshold-based approach. The performance of these two approaches are evaluated in Section VII. Section

VIII concludes the paper.

II. Related Work

A. Time Synchronization Problems and Schemes

Traditional techniques for time synchronization include the Global Positioning System (GPS) [9] and the Network Time Protocol (NTP) [10]. Commercial GPS receivers can synchronize the time to 200 ns. However, GPS services may not be available in places such as inside building and underwater, and it may need several minutes of initial set up time. In addition, GPS units might also be large, power-consuming, and expensive relative to resource constraint sensor nodes. NTP has been widely deployed and proved to be effective, secure and robust in Internet, but not energy efficient. Thus, both protocols are not suitable for wireless sensor networks.

The synchronization method proposed in [11] is the first work addressing the time synchronization issue in sensor networks. In this scheme, the clocks of the sensor nodes are not synchronized. When an event occurs, each node records the time of the event based on its local clock. After that, a third party node, acting as a beacon, broadcasts a synchronization pulse to all nodes in the area. All nodes that have received the pulse will use it as a reference to normalize the event timestamp. This scheme was further extended to the reference broadcast synchronization (RBS) scheme [4]. The RBS scheme however can only synchronize multiple receivers in a local region. Later, a network-wide synchronization scheme, called timing-sync protocol for sensor networks (TPSN)[5], was proposed. In TPSN, all nodes form a hierarchical structure. TPSN works in two phrases. In the first phase, a hierarchical structure is constructed and each node is at a specific level in the hierarchy. In the second phase, a node in level $i+1$ synchronizes with a node in level i. In this way, all nodes in the network synchronize with the root node. Recently, Li and Rus have defined a localized diffusion-based protocol in which nodes achieve synchronization by flooding their neighbors with information about each node's local clock value [12]. After each node has learned the clock values of all its neighbors, the node can use a mutually agreed consensus value to adjust its clock.

Unlike the above schemes [11], [4], [5], [12] which aims at maximizing the accuracy, the lightweight tree-based synchronization (LTS) protocol [6] tries to minimize the complexity of the synchronization. The scheme assumes that there are some reference points which have the accurate time in the network. It is also assumed that the clock drift rates are bounded. Based on these assumptions, two synchronization algorithms are proposed to synchronize nodes in pairwise. The first one is a centralized scheme, where a spanning tree is constructed from the reference point (the root of the tree). Then pairwise synchronization is done from the root to the leaves. The other algorithm is a distributed algorithm in which a node gets synchronized on demand by sending a synchronization request to the reference point. All the nodes along the route will get synchronized.

B. Time Synchronization in Hostile Environments

Most of the aforementioned protocols [11], [4], [5], [6], [12] become vulnerable in hostile environments. Taking the RBS scheme as an example, an attacker may launch different kinds of attacks to break the protocol. The first attack is called *masquerade attack*. Suppose a node A sends out a reference beacon to its two neighbors B and C. An attacker E can pretend to be B and exchange wrong time information with C, disrupting the time synchronization process between B and C. A second attack is called *replay attack*. Using the same scenario in the first attack, the attacker E can replay B's old timing packets, misleading C to be synchronized to a wrong time. A third attack is called *message manipulation attack*. In this attack, an attacker may drop, modify, or even forge the exchanged timing messages to interrupt the time synchronization process. For the message dropping attack, the attacker can selectively drop the packets and thus prolong the converging time of the synchronization process. This can be done on a random or arbitrary basis, making it more difficult to be detected. For the message forging attack, the attacker can forge many reference beacon messages and flood the network. This not only incorrectly synchronizes the neighbors, but also causes those nodes to consume power to process these unwanted and faked timing messages. If some nodes run out of power, coverage holes or network partitions may appear.

We can certainly employ some cryptographic techniques to address the aforementioned attacks. For example, providing authentication of every exchanged message will prevent an outside attacker from impersonating other nodes or altering the content of an exchanged message; while authentication can be achieved using pairwise key pre-distribution schemes such as [13], [14], [15]. Adding a sequence number to beacon messages or other messages will prevent message replay attacks. Message dropping may be noticed by some misbehavior detection schemes [16].

In this paper, we are addressing a new type of attack called *delay attack*, which cannot be prevented or handled by standard cryptography. We will define and discuss the delay attack in Section III.

C. Fault-Tolerance Time Synchronization

The proposed approaches fall into the general field of fault-tolerance time synchronization. This problem has been studied for many years in the past [17], [18], [19], [20]. The algorithms mentioned in [18], [19] are based on an averaging process that involves reading the clocks of all the other processors. Because of the use of averaging, two of the algorithms proposed in [18], [19] requires $3n+1$ processors in order to handle n faults; the third algorithm in [18], [19] requires $2n+1$ processors with the assumption that digital signatures are available. In [20], the nodes are assigned to one or more groups, then each node estimates the clock values of those nodes with which it shares a group. The algorithms in [17] work for arbitrary networks and can tolerate any number of processor or communication link failures as long as the correct processors remains connected by fault-free paths.

Our proposed schemes differ from these schemes in several ways. First, in [18], [19], [20], it was assumed that two nonfaulty clocks never differ by more than a predefined threshold δ, but how to setup this threshold is not discussed. In our solution, we use the time transformation technique to derive the threshold and also give techniques to remove this assumption. Second, [17] requires an authentication mechanism such as digital signatures to ensure that no other node can generate the same message or alter the message without being detected. Our schemes do not have this requirement and can address delay attacks, which can not be handled by cryptographic techniques such as digital signatures, because nodes may be compromised.

III. The Delay Attack Model

The time synchronization schemes proposed for wireless sensor networks are based on two models: the receiver-receiver model or the sender-receiver model. The reference broadcast synchronization scheme (RBS) [4] and its prototype protocol [11] falls into the receiver-receiver model. In the following, we simply use the RBS scheme to represent the receiver-receiver model. Schemes of the sender-receiver model include TPSN [5], LTS [6], and the tiny-sync and mini-sync schemes [7]. In the following, we will describe the delay attack model in the context of the RBS scheme [4].

The RBS scheme is based on a simple idea: using a third party for time synchronization. A node, which is a regular node acting as a *reference* node, broadcasts a reference beacon to its neighbors. Each neighboring node records the arrival time of the beacon based on its own clock. Since these receiving nodes are close to the reference node, we can assume the beacon arrives at both receivers at the same time. Therefore, the difference between the recording times of these receiving nodes is the time offset between them. By exchanging their recorded receiving times, they can calculate the clock offset, adjust and synchronize their clocks. As shown in Figure 1 (a), nodes A and B have the recorded times t_a and t_b, respectively, and the time offset between them is $\delta = t_a - t_b$. To synchronize with node A, node B may increase its clock by δ, or both of them set their clocks to $(t_a + t_b)/2$. The *delay attack* is defined in Definition 1.

Definition 1 (The delay attack): *The attacker deliberately delays some of the time messages, e.g., the beacon message in the RBS scheme, so as to fail the time synchronization process. We refer to this kind of attack as delay attack.*

Figure 1(b) and (c) show two ways to launch the delay attack in the RBS scheme. In Figure 1(b), two colluding nodes act as the reference node for nodes A and B. They send the reference beacon b to nodes A and B at different times. As a result, nodes A and B receive the beacon messages at different times, but they think they receive the beacon at the same time. Figure 1(c) shows that a malicious node can launch the above attacks alone if it has a directed antenna [21] so that nodes A and B only hear one beacon message. The delay attack can also be launched when a benign node is synchronizing with a compromised node. The compromised node can add some delay to the beacon receiving time and send it the good node. This will mislead the good node to synchronize to a wrong time.

The sender-receiver model protocols [5], [6], [7] are also vulnerable to the delay attack. In the sender-receiver model, the sender and the receiver exchange time synchronization packets, estimate the round-trip transmission time between them, and synchronize their clocks after finding the clock offset between them. Since only two nodes are involved in the process, this model does not suffer from the attacks introduced by a malicious reference node. However, a node can be deceived if the node it is synchronizing with is malicious. Therefore, these schemes are also subject to the aforementioned delay attacks.

IV. System Model and Assumptions

A. Node, Network, and Security Assumptions

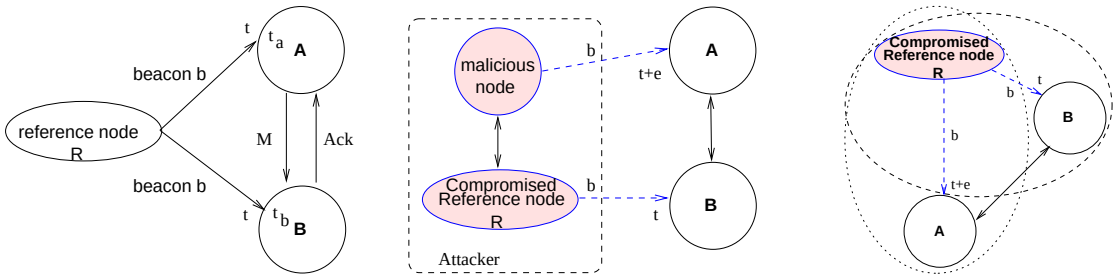
We consider a sensor network composing of resource-constrained sensor nodes such as the current generation of Berkeley Mica motes [22]. Every sensor node is equipped with an oscillator assisted clock and powered by an external battery. The clock of a sensor starts to tick only after it is powered on. Since it is unlikely to power on all the sensor nodes at the same time, there may be large time offsets among sensor nodes initially. We assume that the sensor nodes deployed in a security critical environment is manufactured to sustain possible break-in attacks at least for a short time interval (say several seconds) when captured by an adversary [23]; otherwise, the adversary could easily compromise all the sensor nodes and then take over the network. To this end, we assume that there exists a lower bound on the time interval T_{min} that is necessary for an adversary to compromise a sensor node. We assume that the first time synchronization will be executed and finished within the time interval T_{min}. As a result, we can assume that all the sensor nodes are loosely synchronized.

Because of intrinsic clock drifts of sensor nodes, the time offsets among sensor nodes could become very large (e.g., in the order of seconds or even larger) unless time synchronization is performed once in a while. Hence, we assume that time synchronization is performed periodically. Clearly, the longer the time period, the larger the time offsets. We will discuss the selection of the appropriate synchronization interval in Section VII-C.4.

Each node is assigned a unique id before deployment and it can authenticate the messages sent/received with appropriate shared keys established through a key management protocol [23], [24]. This ensures that no node can impersonate others during the exchange of timing messages and a malicious node can act as a reference at most once.

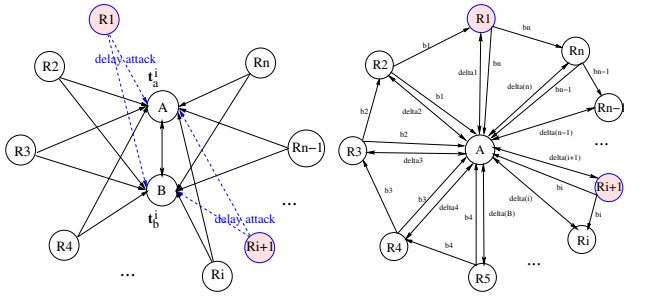
B. Models for Secure Time Synchronization

The general idea of defending against delay attacks is as follows. After collecting a set of time offsets from multiple involved nodes, we identify the malicious time offsets that are under delay attacks. The identified malicious time offsets will be excluded and the rest of the time offsets are used to



(a) The RBS scheme **(b) Collusion-based delay attack** **(c) Directional antenna-based delay attack**

Fig. 1. The RBS scheme and the delay attacks



(a) Two-node model **(b) Neighboring-node model**

Fig. 2. Two models for secure time synchronization

estimate the actual time offset. Next, we present two models for collecting the time offsets: the two-node model and the neighboring-node model, which are described in the context of the RBS scheme.

The two-node model: In this model, one node needs to synchronize with another node. For example, in Figure 2(a), node B is the cluster head and A is a node within the cluster. All nodes in the cluster are required to synchronize with B. Due to security concerns, node A only trusts the cluster head but not other nodes in the cluster. However, it has to use other nodes as reference nodes when using RBS. To deal with security attacks on time synchronization, node A uses multiple reference nodes to obtain a set of time offsets. For example, it can request R_1, R_2,... R_n to serve as reference nodes. Let $\langle t_a^i, t_b^i \rangle$ represent the two beacon receiving times obtained by using a reference node R_i and $\delta_i = (t_a^i - t_b^i)$ be the time offset between A and B. Node B obtains a set of n time offsets $\{\delta_1, \delta_2, \ldots, \delta_n\}$. Based on the collected time offsets, we can detect and exclude the malicious time offsets and obtain a more accurate estimation on the actual time offset between A and B.

The neighboring-node model: In some applications, a node may be required to synchronize with its neighbors to cooperate with each other. In this case, the two-node model is not enough since some neighbors may have been compromised and synchronizing with a malicious node is more vulnerable to attacks. Our solution is illustrated in Figure 2(b). Suppose A has n neighbors: R_1, R_2, …, R_n. We run the RBS scheme between A and each of its neighbors and each time we use a different node as reference to obtain a time offset. After collecting a set of n time offsets, we can detect the outliers, exclude them, and make a good estimation on the actual time offsets.

In addition to the above two models, other models are possible too. However, all of them have one thing in common: they collect a set of time offsets, which may include the malicious time offsets. The focus of this paper is to answer the following question: *Given a set of time offsets, how to identify the outliers and how to achieve an attack-resilient estimation?* In this paper, we propose solutions in the context of RBS, although the solutions can also be applied to the sender-receiver based model.

V. The Outlier-Based Delay Attack Detection

Intuitively, without delay attacks, the time offsets among nodes follow a similar distribution. The existence of delay attacks makes the malicious time offsets much different from the others; otherwise, the attack is not effective and can be tolerated by the time synchronization schemes. In statistics, these malicious time offsets are referred to as *outliers*, which is defined as "an observation which deviates so much from other observations as to arouse suspicious that it was generated by a different mechanism" [25]. Numerous schemes have been proposed to detect outliers [26] (see [26] for a survey). Among them, the generalized extreme studentized deviate many-outlier procedure (GESD procedure) [27] is proved to perform well under different conditions [26]. In the following, we introduce GESD and discuss how to apply it to our problem. After the outliers have been identified by GESD, we discuss how to exclude the outliers and obtain a more accurate estimation of the time offset.

A. The GESD Many-Outlier Detection Procedure

Before introducing GESD, let us first look at the extreme studentized deviate (ESD) test which is also called the Grubb's test. The ESD test is good at detecting one outlier in a random normal sample.

Definition 2 (ESD Test): *Given a data set* $\Gamma = \{x_1, x_2, \ldots, x_n\}$, *The* mean *of* Γ *is denoted as* $\bar{x}$ *and the* standard deviation *of* Γ *is denoted as* s. *Let*

$$T_i = |x_i - \bar{x}|/s, \textit{where } i = 1, \ldots, n.$$

*T_i is also called the corresponding T-*value *of x_i. Let x_j be the observation that leads to the largest $|x_i - \bar{x}|/s$, where $i = 1, \ldots, n$. Then x_j is an* outlier *when T_j exceeds a tabled critical value λ.*

In principle, if T_j does not exceed the critical value λ, we need not single out x_j. Assuming this test finds an outlier, we then look for further outliers by removing observation x_j and repeating the process on the remaining $n-1$ observations. However, the ESD test can only detect one outlier.

The GESD procedure is a modified version of the ESD test, which can find multiple outliers. Two critical parameters for GESD are r and λ_i, where r is the estimated number of outliers in the data set and λ_i is the two-sided $100 * \alpha$ percent critical value got from Formula (1).

$$\lambda_i = \frac{t_{n-i-1,p}(n-i)}{\sqrt{(n-i-1+t^2_{n-i-1,p})(n-i+1)}} \quad (1)$$

In Formula (1), $i = 1, \ldots, r$. $t_{v,p}$ is the $100 * p$ percentage point from the t distribution with v degrees of freedom, and $p = 1 - [\alpha/2(n-i+1)]$. Given α, n and r, the critical values λ_i, where $i = 1, \ldots, r$, can be calculated beforehand.

B. Using GESD for Delay Attack Detection

The GESD-based approach is formally defined as follows.

Definition 3 (Using GESD for delay attack detection): *Given the time offset set $\Gamma = \{\delta_1, \delta_2, \ldots, \delta_n\}$, all the time offsets δ_i that are identified as outliers by GESD are claimed to be under delay attack.*

In GESD, r is the number of estimated outliers in the data set, which is the estimated number of malicious time offsets in our settings. The choice of r plays an important role in GESD. If r is set to a small number and there are more than r malicious time offsets among the n time offsets, some of them cannot be detected using GESD. On the other hand, if r is too large, it wastes time on checking the nodes that are in fact benign (good) ones. In this paper, since the number of time offsets is small (e.g., 20), we set r to be half of the total number of time offsets. We also assume that the number of malicious time offsets is less than half of the total number of time offsets. Without this assumption, GESD may not work since it may find the malicious time offsets to be benign and the benign ones to be malicious.

Definition 4 (Estimate r): *Let the median of the time offset set Γ be $\widehat{x}$ and s be the standard deviation. r is defined as the number of time offsets x_j such that $|x_j - \widehat{x}|/s > 2$, where $i = 1, \ldots, n$.*

When the number of malicious nodes is small, i.e, less than 5% of the total, we can utilize the median of the time offsets to set r. As shown in Definition 4, r is the number of time offsets that are two standard deviations away from the median. In most cases, the data and time offsets are normally distributed, and then 95% of the values are at most two standard deviations away from the mean. In our case, we replace the mean with the median since the median serves better when there exists malicious data sets.

Algorithm 1: Input: r, Γ, λ

```
0   let j = 1, C and T be two arrays
1   begin loop
2       calculate x̄ and s over set Γ; find x_{k_j}
        which maximizes |x_i − x̄|, x_i ∈ Γ;
3       let T[j] = {|x_{k_j} − x̄|/s}, C[j] = x_{k_j};
        remove x_{k_j} from Γ;
4       increase j; decrease r;
5       if (r < 1) break
6   end loop
7   let outlier set Ω = ∅, j = r;
8   begin loop
9       if (T[j] > λ^n[j])
10      then {Ω = ∪{C[k]}, k = 1, ..., j;
                return Ω}
11      else {decrease j; if (j < 1) return ∅}
12  end loop
```

Fig. 3. Identifying outliers with GESD

Figure 3 shows how to use GESD to identify outliers. The algorithm accepts three parameters: the estimated number of outliers r, the time offset data set Γ, and the critical value λ computed by Formula (1). λ can be pre-computed and stored in the sensors. In the following, we use λ^n to denote the critical values for a data set with n elements. Two array structures C and T, are used to save the candidate outlier information. C is used to keep the outliers and T is used to save the T value (Definition 2) corresponding to the candidate outliers. The T values of the candidate outliers are later used to compare with the critical values to decide whether the candidates are outliers or not.

Time complexity In GESD, two operations are time consuming: calculating the mean $\bar{x}$ and the standard deviation s. Among them, the most expensive operation is to calculate the standard deviation, which involves multiplications.

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Given r and n, the first loop (Line 1-6) has $n + (n-1) + \ldots + (n-r) = nr - \frac{1}{2}r^2$ multiplications. In general, the time complexity of GESD is $O(nr)$. In the worst case, where $r = \frac{n}{2}$, the time complexity is $O(\frac{3}{8}n^2)$.

C. Delay Attack Accommodation

The goal of securing time synchronization is to synchronize the time in the presence of delay attacks. This can be achieved by first identifying the outliers (malicious time offsets) and then excluding them when estimating the true time offsets between nodes. We use the mean of the benign time offsets to approximate the true time offsets. The following definition can be used to approximate the time offset estimation $\hat{\delta}$.

Definition 5 (Estimate $\hat{\delta}$): *Let Γ be the time offset data set and Ω be the outlier set. Then the benign time offset set is $\Gamma - \Omega$. $\hat{\delta}$ is defined as the mean of the set $\Gamma - \Omega$. Let the size*

of Γ *be* n *and the size of* Ω *be* k. $\hat{\delta}$ *is calculated as follows.*

$$\hat{\delta} = \sum_{i=1}^{n-k} \frac{x_i}{n-k}, where\ x_i \in \Gamma - \Omega.$$

VI. Threshold-Based Delay Attack Detection

One drawback of the GESD approach is that it needs to have enough reference nodes to detect the malicious nodes effectively. This has been verified by the simulation results shown in Section VII-B. In this section, we propose a threshold-based approach to detect the delay attacks based on the following observations. Without delay attacks, the time offset between two nodes should be bounded by a threshold value if the maximum clock drift rates can be bounded. With the threshold value, we can identify those time offsets that are larger than the threshold as malicious ones. Different from GESD, the threshold-based approach does not need that many reference nodes. Moreover, the threshold-based approach only needs to calculate the threshold once, and hence has less overhead.

In the following, we first present the time transformation technique and then present a method to determine the threshold based on the time transformation technique. After determining the threshold, we discuss how to use it to defend against delay attacks. Different from the previous work [28] where the time interval is used to order messages, we utilize the time interval to quantify the time offset upper bound between two nodes.

A. The Time Transformation Technique

Before presenting the time transformation technique, let us first look at the hardware oscillator assisted clock in Berkeley Mica motes [22], which implements an approximation $C(T)$ of the actual time T. $C(T) = k\int_{T_0}^{T} \omega(\eta)d_\eta + C(T_0)$ is a function of the real time T, which derives from the angular frequency $\omega(T)$ of the hardware oscillator. In this formula, k is a proportional coefficient and T_0 is the initial clock value.

For a perfect hardware clock, $\frac{d_C}{d_T}$ is equal to one. However, all hardware clocks are not perfect since they are subject to *clock drift*. We can only assume that the clock drift rate of the sensor clock does not exceed the maximum value of ρ. Thus, we have the following inequality: $1-\rho \le \frac{d_C}{d_T} \le 1+\rho$.

The idea of time transformation is to transform the real time difference Δ_T into the sensor clock difference Δ_C and vice versa. These transformations are difficult because of the unpredictability of the sensor clock, but there exists some lower and upper bounds on the estimates. Based on the previous inequality, we can get: $1-\rho \le \frac{\Delta_C}{\Delta_T} \le 1+\rho$. This inequality can be transformed into $(1-\rho)\Delta_T \le \Delta_C \le (1+\rho)\Delta_T$ and $\frac{\Delta_C}{1+\rho} \le \Delta_T \le \frac{\Delta_C}{1-\rho}$, which means that the clock difference Δ_C can be approximated by the interval $[(1-\rho)\Delta_T, (1+\rho)\Delta_T]$. On the other hand, the real time difference Δ_T that corresponds to the sensor clock difference Δ_C can be approximated by the interval $[\frac{\Delta_C}{1+\rho}, \frac{\Delta_C}{1-\rho}]$.

In order to transform a time difference Δ_{C_1} corresponding to one node N_1 with ρ_1, to a time corresponding to another node N_2 with ρ_2, Δ_{C_1} is first estimated by the real time interval $[\frac{\Delta_{C_1}}{1+\rho_1}, \frac{\Delta_{C_1}}{1-\rho_1}]$, which in turn is estimated by the sensor clock time interval $[\frac{1-\rho_2}{1+\rho_1}\Delta_{C_1}, \frac{1+\rho_2}{1-\rho_1}\Delta_{C_1}]$, relative to the local time of node N_2. As shown in Figure 4, nodes A and B use RBS to do time synchronization. The maximum clock drift rates of A and B are denoted as ρ_a and ρ_b, respectively. Suppose A and B receive the reference beacon at time t_a and t_b, in terms of their own local clocks, respectively. After receiving the reference beacon, at time t_1, A sends a message M to B, telling B that it received the beacon at time t_a. Message M is received by B at time t_3, and then B sends back an Ack at time t_4 to confirm that it has received M. In the Ack, B piggybacks t_b, t_3, and t_4. After receiving the Ack, A can use the time transformation technique to transform the beacon receiving time t_b to a time interval $[t_{bL}, t_{bR}]$ relative to A's clock as follows.

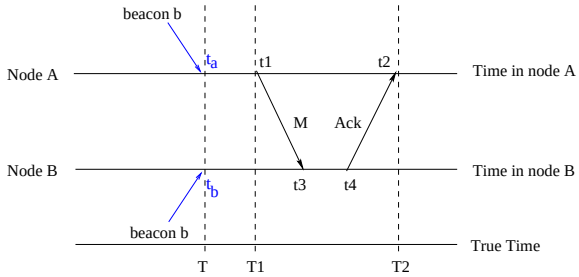


Fig. 4. Time transformation

$$\begin{aligned} t_{bL} &= t_2 - (t_4 - t_b)\tfrac{1+\rho_a}{1-\rho_b} - ((t_2 - t_1) - (t_4 - t_3)\tfrac{1-\rho_a}{1+\rho_b}) \\ t_{bR} &= t_2 - (t_4 - t_b)\tfrac{1-\rho_a}{1+\rho_b} \end{aligned} \quad (2)$$

B. Determining the threshold ξ

The threshold ξ is the upper bound of the time offsets between two nodes. We determine ξ based on the idea of time transformation shown above. A straightforward solution is to use $(t_{bR} - t_{bL})$ as ξ. However, $(t_{bR} - t_{bL})$ is a tight bound. If we use it to decide whether a time offset is malicious or not, it may identify benign time offsets as malicious. To effectively detect malicious time offsets, ξ should be a looser upper bound. Since t_{bL} and t_{bR} are the two boundaries of time t_b at node A, $max(|t_a - t_{bL}|, |t_{bR} - t_a|)$ should be the upper bound of the time offsets between A and B. Based on this observation, the time offset upper bound, ξ^{ab}, between A and B can be determined by Formula (3), which is a looser upper bound compared to $(t_{bR} - t_{bL})$. This can be explained as follows. If the clock drift rate of the two nodes are equal, t_a should fall inside $[t_{bL}, t_{bR}]$; otherwise, t_a may fall outside of $[t_{bL}, t_{bR}]$, leading to a looser upper bound based on Formula (3). Since the clock drift rates of two nodes are usually not equal, Formula (3) gives a looser upper bound compared to $(t_{bR} - t_{bL})$.

$$\xi^{ab} = \begin{cases} t_{bR} - t_a & \text{if } t_a < t_{bL} \\ \text{MAX}\{t_{bR} - t_a, t_a - t_{bL}\} & \text{if } t_a \in [t_{bL}, t_{bR}] \\ t_a - t_{bL} & \text{if } t_a > t_{bR} \end{cases} \quad (3)$$

The time offset upper bound between two neighboring nodes shown in Formula (3) is calculated only in the first time synchronization, which happens shortly after the sensor network deployment. Thus, the time offset caused by the clock drift is small in Formula (3). The clock drift time increases as time goes by. If the time synchronization interval is long, the clock drift time will be long and should be taken into consideration when determining the time offset upper bound.

Formula (4) gives the time offset upper bound between nodes A and B considering clock drift time.

$$\Delta^{ab} = \xi^{ab} + |\rho_a - \rho_b| \cdot T \tag{4}$$

In Formula (4), T is the time synchronization interval and Δ^{ab} is the upper bound of the time offset between nodes A and B when they are synchronized using one reference node. To increase the accuracy of the estimation, we use n reference nodes to obtain a set of ξ^{ab}. The threshold ξ is defined as the maximum among them, as showed in Formula (5).

$$\xi = \text{MAX}\left\{\xi_i^{ab}\right\} + |\rho_a - \rho_b| \cdot T, \text{where } 1 \leq i \leq n. \tag{5}$$

With threshold ξ, we can detect malicious time offsets among a set of time offsets. The threshold-based approach is formally defined in Definition 6.

Definition 6 (The threshold-based delay attack detection): Given the time offset data set $\Gamma = \{\delta_1, \delta_2, \ldots, \delta_n\}$, all the time offsets bigger than ξ are claimed to be under delay attack and are identified as malicious time offsets.

Time complexity Compared to GESD, the threshold-based approach involves two multiplications when calculating the interval $[t_{bL}, t_{bR}]$ in Formula (2). Given n reference nodes, the total number of multiplications are $2n$. Thus, the time complexity of the threshold-based approach is $O(2n)$, which is much less than that of the GESD approach, which is $O(nr)$. Further, the threshold is only calculated in the first time synchronization, but the GESD outlier detection algorithm is executed for each time synchronization. Thus, the GESD approach has much higher overhead than the threshold based approach.

C. Delay Attack Accommodation

After the malicious time offsets have been detected using the threshold, we can use the same strategy as that in Section V-C to exclude them and obtain a good estimation on the true time offset between two nodes.

VII. Performance Evaluations

A. Simulation setup

We evaluate the performance of the two approaches using the reference broadcast synchronization (RBS) scheme by simulation. In the simulation, each node has a maximum clock drift rate at microsecond level (10^{-6} second) [28]. The deviations of clock drift rates among nodes are also at microsecond level. To synchronize two nodes, a number of reference nodes are generated varying from 10 to 20. Each reference node broadcasts a reference beacon to these two nodes, which record the beacon receiving times according to their clocks. The arrival times of the reference beacons follow Poisson distribution, and the beacon processing time follows normal distribution. Since the typical message size is 36 bytes in TinyOS [29], the beacon processing time is about 12 milliseconds which is the time required to process a 36-byte packet.

After a beacon has been processed, one node sends the beacon receiving time to the other, which calculates the time offset between them. After these two nodes get a set of time offsets, we randomly pick some of them as malicious time offset and assume they are under delay attacks. We also add a delay attack time which follows normal distribution. Based on a set of time offsets, the proposed schemes are evaluated with different levels of delay attack time and different number of malicious time offsets. All results are obtained by setting the synchronization interval to 5,000 seconds. The results are averaged over 100 runs. Most of the simulation parameters are listed in Table I.

Number of reference nodes	10 to 20
Number of malicious nodes	1 to 5
Beacon processing time mean	12 milliseconds
Beacon arrival interval mean	200 milliseconds
Clock drift rate mean	0.005 millisecond
Clock drift rate deviation	0.001 millisecond
Delay attack time	1 - 100 milliseconds
Synchronization interval	5,000 seconds

TABLE I
SIMULATION PARAMETERS

Three metrics are used to evaluate the effectiveness of the proposed schemes: the successful detection rate, the false positive rate, and the accuracy improving rate. In a network with delay attacks, the successful detection rate tells the percentage of malicious time offsets that can be successful detected. The false positive rate shows the percentage of time offsets that are reported as outliers but are not. The accuracy improving rate shows the accuracy improvement on the estimated time offset after the detected outliers have been excluded. Let $\hat{\delta}$ be the estimated time offset when the outliers have been excluded and δ_{bad} be the estimated time offset when the outliers have not been excluded. The accuracy improving rate is defined in Formula (6).

$$\text{Accuracy improving rate} = \frac{\delta_{bad} - \hat{\delta}}{\hat{\delta}} * 100\% \tag{6}$$

B. Simulation Results of the GESD Approach

1) The Successful Detection Rate: Figure 5 shows the successful detection rate as the delay attack time (*delay*), the number of malicious nodes, and the number of time offsets (NUM_REF) change. We did not show the successful detection rate when there are five malicious nodes for NUM_REF=10, because GESD does not work when the number of malicious time offsets is equal or larger than that of the benign nodes.

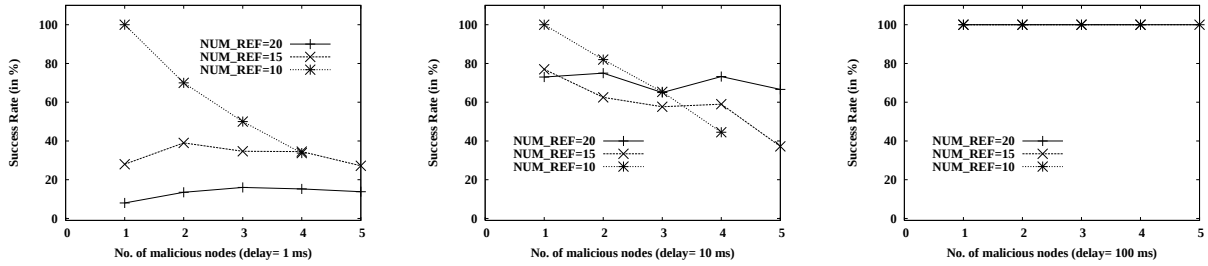


Fig. 5. The successful detection rate of GESD

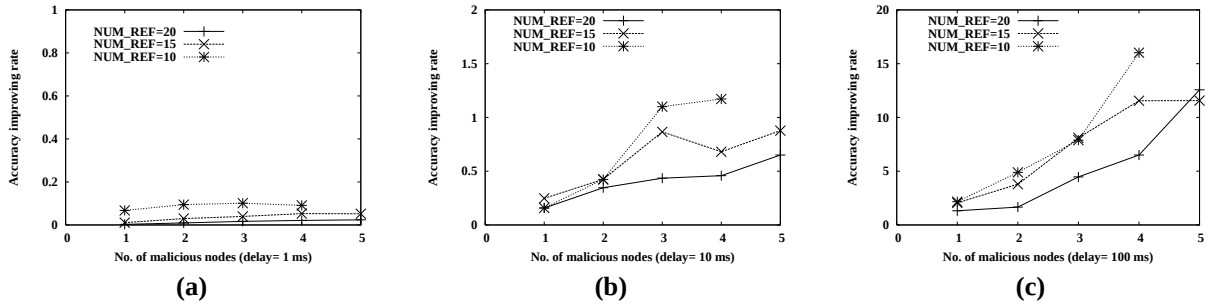


Fig. 6. The accuracy improving rate of GESD

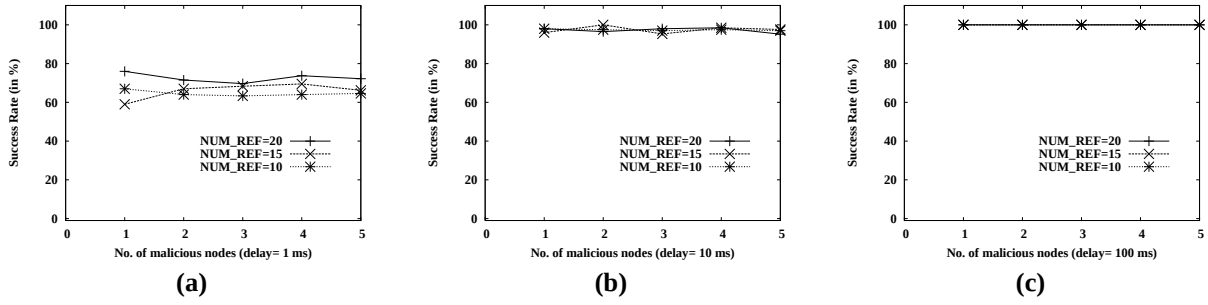


Fig. 7. The successful detection rate of the threshold-based approach

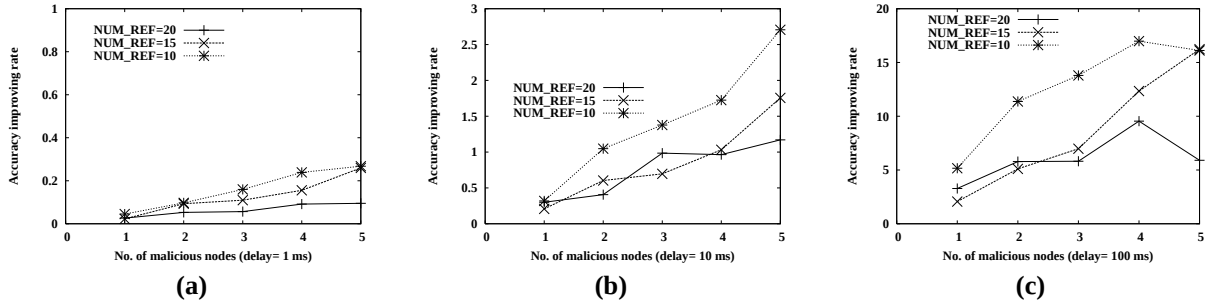


Fig. 8. The accuracy improving rate of the threshold-based approach

Based on the figure, we have the following observations. First, when the delay attacks are at levels of 1ms 10ms, the successful detection rate is pretty low in most cases. Since the time synchronization interval is 5000 seconds, the clock drift time between two nodes can be as large as 10ms. It is difficult to detect the delay attacks when the delay attack time is not significantly larger than the clock drift time, resulting in low successful detection rate. In Figure 5(a), since the delay attack time is one level smaller than the clock drift time, increasing the number of reference nodes does not help improving the successful detection rate.

Second, as shown in Figure 5(b), when the number of malicious time offsets increases, the successful detection rate decreases due to the masking problem in outlier detection. Masking occurs when an outlier goes undetected because of the presence of other outliers. GESD is not robust against the masking problem since it is based on the *mean* value, which is affected by the outliers. As an exception, when NUM_REF is 20 and the number of malicious time offsets increases from three to four, the successful detection rate increases. This can

be explained as follows. If one malicious time offset is not detected in both cases, the successful detection rate will be about 66% when the number of malicious time offsets is three and 75% when the number of malicious time offsets is four, which shows an increase in terms of successful detection rate.

Third, Figure 5(b) also shows that the successful detection rate increases as the number of time offsets increase in general. Given a number of malicious time offsets, we will have more benign time offsets with a larger set of time offsets; and the more benign nodes we have, the higher the successful detection rate is. Thus, when there are multiple outliers, GESD is more effective if more time offsets are available.

Fourth, as long as the delay attack time is much larger than the clock drift during the synchronization interval, the successful detection rate increases dramatically. For example, as shown in Figure 5 (c), the successful detection rate reaches 100% when the delay attack is at 100ms level. As the delay attack time is larger than the clock drift time, the malicious time offsets can be easily identified. Although not shown in the figure, GESD keeps the 100% successful detection rate when the the delay attack time is larger than 100ms.

2) The False Positive Rate: The simulation results show that the false positive rate of GESD is almost 0 in our system settings. This is because a benign time offset will not be identified as outlier when there really exists malicious nodes. Thus, GESD works well in terms of false positive rate.

3) The Accuracy Improving Rate: Figure 6 shows the accuracy improving rates with different level of delay attacks. From the figure, we can see that the accuracy improving rate is low when the delay attacks are at levels of 1ms and 10ms. This is because the delay attack time is relatively small compared to the clock drift time during the 5000-second interval. Thus, excluding the malicious time offsets cannot have too much improvement. However, as the delay attack time increases, excluding the malicious time offsets can significantly improve the accuracy improvement rate. For example, when the delay attack time is 100ms, the accuracy improving rate can be increased by as much as 16 times (see Figure 6(c)).

C. Simulation Results of the Threshold-based Approach

1) The Successful Detection Rate : Figure 7 shows the successful detection rates with different level of delay attacks when the synchronization interval is 5000-second. As shown in Figure 7(a) and (b), when the delay attack time is 1ms, the threshold-based scheme can achieve a higher successful detection rate compared to GESD (Figure 5(a)(b)). For example, when NUM_REF is 20, the successful detection rate of the THRESHOLD-based approach (80% on average) is seven times higher than that of GESD (10% on average). This shows that the threshold-based approach is effective even when the delay attack time is small compared to the clock drift rate. In the threshold-based approach, the threshold reflects both the maximum time offset that two nodes can have when there is no delay attack and the time offset caused by clock drift during the synchronization interval. Thus, even though delay attack time is not large compared to the clock drift time, it can still be detected at a high rate. Similar to GESD, the threshold-based approach achieves a 100% successful detection rate when the delay attack time is 100ms.

Figure 7 also shows that the successful detection rate does not change too much as the number of malicious time offsets increases. Different from GESD, the threshold is affected neither by the outlier masking problem nor by the number of malicious time offsets.

In summary, the threshold-based approach can achieve a better successful detection rate than GESD. The threshold-based scheme performs well even when the delay attack time is small compared to the clock drift time and it is robust against multiple delay attacks.

2) The False Positive Rate: Simulation results show that the false positive rate of the threshold-based approach is always 0 in different settings. This is due to the reason that the threshold is determined in such a way that no benign time offsets will be identified as malicious. From the false positive rate point of view, both the GESD approach and the threshold-based approach perform well.

3) The Accuracy Improving Rate: Figure 8 shows the accuracy improving rates with different level of delay attacks. As shown in Figure 8 (a), when the delay attack time is 1ms, the accuracy improving rate is below 30% most of time, because the delay attack time is small compared to the clock drift time. However, the accuracy improving rate achieved in the threshold-based approach is much higher than that of GESD. This can be explained by the fact that the threshold-based approach can achieve a much higher successful detection rate than GESD. As the delay attack time increases, the improvement on the accuracy also increases as shown in Figure 8(b) and (c). In terms of the accuracy improving rate, the threshold-based approach performs better than GESD, which is consistent with the results of the successful detection rate.

4) The Synchronization Interval: Figure 9 shows the impact of the synchronization interval on the successful detection rate under different level of delay attacks. The results are obtained when the number of time offsets is 10. As shown in the figure, given a certain level of delay attack, the successful detection rate decreases as the synchronization interval increases. For example, when the delay attack time is 10ms, the threshold-based approach can almost reach 100% detection rate when the synchronization interval is less than 3,000 seconds. When the synchronization interval is larger than 50,000 seconds (or 13.9 hours), the successful detection rates drops to about 60% on average.

Figure 10 shows the tolerable synchronization interval with different levels of delay attacks. We define the *tolerable synchronization interval* as the maximum synchronization interval with which the threshold-based scheme achieves a 99% or higher successful detection rate. Here, we still use ten reference nodes. We observe that the tolerable synchronization interval increases as the delay attack time increases. Figure 10 shows that the tolerable synchronization intervals are 0.83, 7.78, 83.33, 888.89, and 8055.56 hours when the delay attacks

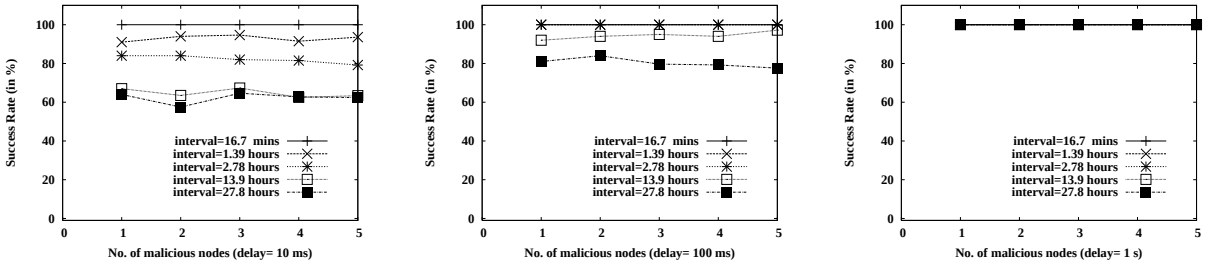


Fig. 9. Delay attacks and the synchronization interval

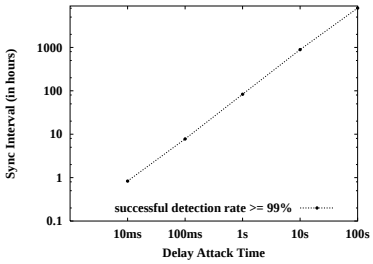


Fig. 10. The tolerable synchronization interval

are 10ms, 100ms, 1s, 10s, and 100s respectively. Thus, given a delay attack time, we can select the appropriate synchronization interval without sacrificing the successful detection rate. Since the threshold-based approach can tolerate 83.33-hour synchronization interval on 1-second level delay attacks, we claim that our threshold-based scheme is very effective on defending against delay attacks in sensor networks.

VIII. Conclusions

In this paper, we identified various attacks that are effective to several representative time synchronization schemes, and focused on dealing with the delay attack. We proposed two solutions to detect and accommodate the delay attacks. Our first approach uses the generalized extreme studentized deviate (GESD) algorithm to detect multiple outliers introduced by the compromised nodes and our second approach uses a threshold derived using a time transformation technique to filter out the outliers. Extensive simulation results show that both schemes are effective in defending against delay attacks. However, the GESD approach needs more reference nodes to effectively detect the malicious nodes. The threshold based approach relaxes this assumption and outperforms GESD in terms of successful detection rate, false positive rate, and accuracy improving rate.

References

[1] Wensheng Zhang and Guohong Cao, "Optimizing tree reconfiguration for mobile target tracking in sensor networks," *IEEE INFOCOM*, 2004.

[2] Wensheng Zhang and Guohong Cao, "Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor network," *IEEE Transactions on Wireless Communication*, vol. 3, no. 5, pp. 1689–1701, 2004.

[3] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Network and Distributed System Security Symposium, NDSS '01*, Feb. 2001, pp. 35–46.

[4] Jeremy Elson, Lewis Girod, and Deborah Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[5] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. 2003, pp. 138–149, ACM Press.

[6] Jana van Greunen and Jan Rabaey, "Lightweight time synchronization for sensor networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. 2003, pp. 11–19, ACM Press.

[7] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," *Wireless Communications and Networking (WCNC'03), IEEE*, vol. 2, pp. 16–20, March 2003.

[8] M. Chen, W. Cui, V. Wen, and A. Woo, "Security and deployment issues in a sensor network," 2000.

[9] E. D. Kaplan, editor, *Understanding GPS: Principles and Applications*, Artech House, 1996.

[10] David L. Mills, "Internet time synchronization: The network time protocol," in *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*, 1994.

[11] Jeremy Elson and Deborah Estrin, "Time synchronization for wireless sensor networks," in *Proceedings of the 15th International Parallel & Distributed Processing Symposium*. 2001, p. 186, IEEE Computer Society.

[12] Qun Li and Daniela Rus, "Global clock synchronization in sensor networks," in *IEEE Infocom 2004*, Hong Kong, china, March 2004.

[13] Laurent Eschenauer and Virgil D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, New York, NY, USA, 2002, pp. 41–47, ACM Press.

[14] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, Washington DC, October 27-31 2003.

[15] Haowen Chan, Adrian Perrig, and Dawn Song, "Random key predistribution schemes for sensor networks," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2003, p. 197, IEEE Computer Society.

[16] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000, pp. 255–265, ACM Press.

[17] Joseph Y. Halpern, Barbara Simons, Ray Strong, and Danny Dolev, "Fault-tolerant clock synchronization," in *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, New York, NY, USA, 1984, pp. 89–102, ACM Press.

[18] Leslie Lamport and P. M. Melliar-Smith, "Byzantine clock synchronization," in *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, New York, NY, USA, 1984, pp. 68–74, ACM Press.

[19] Leslie Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, pp. 52–78, 1985.

[20] A. Olson and K. G. Shin, "Fault-tolerant clock synchronization in large multicomputer systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 9, pp. 912–923, 1994.

[21] C. Santivanez and J. Redi, "On the use of directional antennas for sensor networks," in *Military Communications Conference (MILCOM 2003)*, , October 2003.

[22] Crossbow Technology Inc., "Wireless sensor networks," in *http://www.xbow.com/*, Accessed in November, 2004.

[23] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM conference on Computer and communications security*. 2003, pp. 62–72, ACM Press.

[24] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler, "Spins: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002.

[25] D. M. Hawkins, *Identification of Outliers*, New York: Chapman and Hall, 1980.

[26] B. Iglewicz and D. C. Hoaglin, *How to Detect and Handle Outliers*, ASQC basic references in quality control, 1993.

[27] B. Rosner, "Percentage points for generalized esd many-outlier procedure," *Technometrics*, 1983.

[28] Kay Römer, "Time synchronization in ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. 2001, pp. 173–182, ACM Press.

[29] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System architecture directions for networked sensors," *SIGOPS Oper. Syst. Rev.*, vol. 34, no. 5, pp. 93–104, 2000.