Secure and Efficient Time Synchronization in Heterogeneous Sensor Networks

Xiaojiang Du, *Member, IEEE*, Mohsen Guizani, *Senior Member, IEEE*, Yang Xiao, *Senior Member, IEEE*, and Hsiao-Hwa Chen, *Senior Member, IEEE*

***Abstract*—Due to the collaborative nature of sensor nodes, time synchronization is critical for many sensor network operations. For sensor networks deployed in hostile environments, security is critical to the success of time synchronization. Over the past few years, a number of secure time-synchronization schemes have been proposed for sensor networks. However, these schemes are designed for homogeneous sensor networks and may incur large communication/computation overhead or may cause accumulated synchronization errors (due to multihop relays of time reference messages). Several works have shown that better performance and security can be achieved in heterogeneous sensor networks (HSNs). In this paper, we present a secure and efficient time synchronization scheme for HSNs by utilizing powerful high-end sensors. We implement the time-synchronization scheme in real sensor nodes, and the experiments show that our scheme achieves higher synchronization accuracy than a popular sensor time-synchronization scheme. The analyses demonstrate that our scheme is resilient to various attacks and significantly reduces communication overhead.**

***Index Terms*—Heterogeneous sensor networks (HSNs), security, time synchronization.**

I. Introduction

DUE TO THE collaborative nature of sensor nodes, time synchronization is very important for many sensor network operations, such as coordinated sensing tasks, sensor scheduling (sleep and wake) [1], [2], mobile object tracking [3], [4], time-division multiple-access medium access control, data aggregation [5], multicast source authentication protocol [6], and so on. For example, in a target-tracking application, sensor nodes need to know both the location and the time when the target is sensed to correctly determine the moving target's direction and speed.

Several time-synchronization algorithms (e.g., [7]–[9]) have been proposed for sensor networks. However, none of the above time synchronization schemes was designed with security in mind. Hence, they are not suitable for applications in hostile environments (such as a military battlefield and a security monitoring area), where security is critical. Most existing time synchronization schemes are vulnerable to several attacks. In [10], Song *et al.* identify the following four possible attacks.

1) *Masquerade attack*: Suppose that a node A sends out a reference beacon to its two neighboring nodes B and C. An attacker E can pretend to be B and exchange wrong time information with node C, disrupting the time synchronization process between nodes B and C.
2) *Replay attack*: Using the same scenario in the first attack, attacker E can replay node B's old timing packets, misleading node C to be synchronized to a wrong time.
3) *Message manipulation attack*: In this attack, an attacker may drop, modify, or even forge the exchanged timing messages to interrupt the time synchronization process.
4) *Delay attack*: The attacker deliberately delays some of the time messages, e.g., the beacon message in the reference-broadcast synchronization (RBS) [7] scheme, to fail the time-synchronization process. Note that this attack cannot be defended by cryptographic techniques.

The first three attacks can be addressed by cryptographic techniques. Authentication can be used to defend against the *masquerade attack*. To prevent the *replay attack*, a sequence number can be added to each exchanged message. Message dropping may be noticed by some misbehavior detection schemes [11]. However, the *delay attack* and *denial-of-service (DoS) attack* cannot be defended by cryptographic techniques. In [10], Song *et al.* identify the *delay attack* and propose solutions to defend the attack. In [12], Sun *et al.* propose a secure clock synchronization scheme based on broadcasts. However, the scheme in [12] has the following two limitations: 1) It requires that the nodes in a cluster maintain initial synchronization, and 2) it requires that each node should be able to reach all the other nodes in a cluster. In [21], Hu *et al.* propose secure time synchronization for underwater sensor networks.

The existing time synchronization schemes for homogeneous sensor networks (including the schemes in [10] and [12]) involve nontrivial computation and communications and, thus, incur large overhead. Furthermore, many synchronization algorithms need to propagate a time synchronization message from some reference point [e.g., the base station (BS)] to all sensors via multihops, and a synchronization error is accumulated during the multihop transmissions.

To improve performance and security, we adopt a heterogeneous sensor network (HSN) model, which consists of a small number of powerful high-end sensors (H-sensors), e.g.,

Manuscript received June 1, 2006; revised October 31, 2006 and January 9, 2007. This work was supported by the U.S. National Science Foundation and the Army Research Office. The review of this paper was coordinated by Prof. X. Zhang.

X. Du is with the Department of Computer Science, North Dakota State University, Fargo, ND 58105 USA (e-mail: dxj@ieee.org).

M. Guizani is with the Department of Computer Science, Western Michigan University, Kalamazoo, MI 49008 USA (e-mail: mguizani@ieee.org).

Y. Xiao is with the Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487 USA (e-mail: yangxiao@ieee.org).

H.-H. Chen is with the Department of Engineering Science, National Cheng Kung University, Tainan 701, Taiwan, R.O.C. (e-mail: hshwchen@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2007.912327

personal digital assistants, and a large number of low-end sensors (L-sensors), e.g., Motes. The use of heterogeneous nodes in sensor networks is not new. Recently deployed sensor network systems are increasingly following heterogeneous designs, incorporating a mixture of sensors with widely varying capabilities. A few works (such as [13]–[15]) have studied the various nonsecurity aspects of HSNs.

The goals of this paper are given as follows: 1) to propose an efficient and effective time-synchronization scheme for HSNs, which has better accuracy and much less communication overhead than existing schemes and 2) to design two authentication methods to provide security for the time-synchronization scheme.

The rest of the paper is organized as follows. In Section II, we briefly describe the clustering scheme in HSNs and the assumptions about HSNs. In Section III, we present the secure time synchronization scheme for HSNs. In Section IV, we evaluate the accuracy of the time-synchronization scheme via real sensor experiments and analyze the security and communication overhead of the scheme. We provide the conclusions in Section V.

II. HSN Model

We adopt a realistic model of an HSN that can be applied to most sensor network applications. In the HSN model, both H- and L-sensors are powered by batteries and have limited energy and communication capabilities. L-sensors use multihop communications to reach H-sensors, and H-sensors use multihop communications to reach a BS. Compared to an L-sensor, an H-sensor has a larger transmission range (power), better computation capability, larger storage, more energy supply, and better reliability. By utilizing the powerful H-sensors, we design an efficient and secure time synchronization scheme for HSNs. The scheme can defend against all the attacks mentioned in Section I, and it incurs small communication and computation overhead for L-sensors. The details of the scheme are presented in Section III.

Both L- and H-sensors are distributed in the network. Note that our time-synchronization scheme does not rely on any sensor distribution pattern. For the ease of discussion below, we assume that each H-sensor can directly communicate with its neighbor H-sensors (if not, then relays via L-sensors can be used). All H-sensors form a backbone in an HSN. We have designed an efficient and robust clustering scheme for HSNs in [15]. Due to the page limit, we will not describe the scheme here. After cluster formation, an HSN is divided into multiple clusters, where each H-sensor serves as the cluster head. We list the assumptions of HSNs as follows.

1) Due to cost constraints, L-sensors are not equipped with tamper-resistant hardware. Assume that if an adversary compromises an L-sensor, it can extract all key materials, data, and codes that are stored in that node.
2) Each L-sensor (and H-sensor) is static and aware of its own location. Sensor nodes may use a secure localization algorithm (such as the one in [16]) to estimate their locations, and no GPS receiver is required at each node.
3) Each L-sensor (and H-sensor) has a unique node identification (ID).
4) H-sensors are equipped with tamper-resistant hardware. Tamper-resistant hardware is too expensive for L-sensors. However, it is reasonable to assume that powerful H-sensors are equipped with the technology. In addition, the number of H-sensors in an HSN is small (e.g., 20 H-sensors and 1000 L-sensors in an HSN). Hence, the total cost of tamper-resistant hardware is acceptable.
5) The BS is trusted.

III. Secure Time-Synchronization Scheme

In this section, we present an efficient and secure time synchronization scheme for HSNs. In an HSN, clusters are formed, and H-sensors serve as cluster heads. An H-sensor has a long transmission range and can reach almost all L-sensors in its cluster by one broadcast. Broadcasts from an H-sensor can be utilized for efficient and secure time synchronization for nodes in a cluster. An H-sensor can include its time in a broadcast message, and then, every L-sensor synchronizes its clock with the H-sensor. The time synchronization in an HSN consists of two steps. First, all H-sensors are synchronized (with the BS). Then, each L-sensor synchronizes the clock with its cluster head (an H-sensor). We present the two parts in Sections III-A and B, respectively.

A. Synchronizing H-Sensors

There are different approaches to synchronize H-sensors in an HSN. Because H-sensors are powerful high-end nodes, it is possible that they are installed with global positioning system (GPS) receivers (for other purposes such as localization). In such a case, time synchronization for H-sensors is achieved by the GPS service. Note that because the number of H-sensors (e.g., 50) in an HSN is small, the total cost of GPS receivers is also small. If the GPS service is not available to H-sensors, either because of bad environment (e.g., underwater) or no GPS receivers are installed (due to cost constraint), many existing time-synchronization algorithms can be modified to provide secure time synchronization for H-sensors.

In this section, we present a secure time synchronization scheme for H-sensors, which is based on a sender–receiver scheme such as the timing-sync protocol for sensor networks (TPSN) [8]. The basic idea of TPSN is to use a two-way message exchange between a pair of nodes. As shown in Fig. 1, at time T1, node A sends out a message $m1$, at T2, node B receives message $m1$, at T3, node B sends out a message $m2$, and at T4, node A receives message $m2$. Note that T1 and T4 are local timestamps of node A, and T2 and T3 are local timestamps of node B. Denote Δ and d as the clock drift between the two nodes and the propagation delay, respectively. Assuming that the clock drift and the propagation delay do not change in this short span of time, then we have $\mathrm{T2} = \mathrm{T1} + \Delta + d$ and $\mathrm{T4} = \mathrm{T3} - \Delta + d$. Thus, $\Delta = [(\mathrm{T2} - \mathrm{T1}) - (\mathrm{T4} - \mathrm{T3})]/2$ and $d = [(\mathrm{T2} - \mathrm{T1}) + (\mathrm{T4} - \mathrm{T3})]/2$. By knowing the drift, node A can correct its clock accordingly so that it synchronizes to node B. See [8] for more details about TPSN.

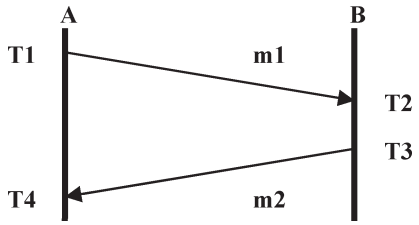

Fig. 1. TPSN.

The original TPSN [8] is not a secure time synchronization scheme. In this paper, we present a secure TPSN scheme that can prevent all of the four attacks on the time synchronization process. To provide security to TPSN, we basically need to provide protection to the exchanged messages. We refer to the exchanged messages as timing messages. First, a timing message needs to include the sender's time (e.g., T1). For security purposes, the sender (e.g., A) includes a message sequence number and then calculates a message authentication code (MAC) by using the shared key between nodes A and B. Thus, the timing message has the following format: Timestamp + Sequence number + MAC. The secure TPSN can detect all of the four attacks on time synchronization, except for the message drop attack. An attacker E cannot pretend to be node A and send a valid timing message to node B because node E does not know the shared key between nodes A and B, and it cannot generate a valid MAC. Thus, the *masquerade attack* is prevented. Because each timing message includes a different message sequence number, an attacker cannot replay an old timing message, and thus, the *replay attack* is prevented. Again, because an attacker E does not know the shared key between nodes A and B, it cannot modify or forge a timing message without being detected, so the *message manipulation attack* (except the message drop attack) is prevented. In TPSN, the timing message is only exchanged between two neighbor nodes. Thus, the *delay attack* cannot be launched, i.e., once node A sends out message $m1$, an attacker E cannot delay the arrival of the message to node B. Note that an attacker E may launch a jamming attack to corrupt message $m1$ such that node B could not receive a correct message. However, because the defense against a jamming attack is out of the scope of this paper, we will not discuss such a scenario here. Next, we want to discuss the defense against the message drop attack. The message drop attack would not be effective on TPSN, and the reason is the same as that for the *delay attack*, i.e., in TPSN, each timing message is directly sent to a one-hop neighbor, and thus, there is no chance for an attacker to be an intermediate node and to forward timing messages to other nodes. To sum up, the enhanced secure TPSN can defend against all of the four attacks mentioned in Section I.

B. Synchronizing L-Sensors

After all H-sensors have been synchronized, the next step is to synchronize L-sensors in a cluster with the cluster head (an H-sensor). The scheme (without security) of synchronizing L-sensors is simple. An H-sensor has a long transmission range and can reach almost all L-sensors in its cluster by one broadcast. Each H-sensor broadcasts a timing message, and then, every L-sensor in the cluster synchronizes its clock with the H-sensor. The propagation delay is estimated based on the locations of the H- and L-sensors. The H-sensor broadcasts its location to all L-sensors in the cluster, and then, each L-sensor can estimate the propagation delay. Sensor nodes may obtain their location information via some secure location discovery services such as the one in [16].

Assume that an H-sensor broadcasts a timing message at time T1 (H-sensor's local time) and that an L-sensor receives the message at time T2 (L-sensor's local time). Denote Δ and d as the clock drift between the two nodes and the propagation delay, respectively. Similar to TPSN, we have $\mathrm{T2} = \mathrm{T1} + \Delta + d$. There are two unknown variables in the above equation, i.e., Δ and d. In TPSN, two messages are exchanged to obtain the clock drift. In our scheme, the propagation delay is estimated by the location information, and then, only one message is required to estimate the clock drift. The uncertainty at the sender side includes the following two parts: 1) the time spent at the sender to construct the message and 2) medium access control delay at the sender. The result in [8] shows that the uncertainty at the sender side contributes little to the synchronization error (e.g., 0.62 μs to an average error of 16.9 μs), so it can be neglected. Thus, in our scheme, the clock drift between an L-sensor and its cluster head is $\Delta = \mathrm{T2} - \mathrm{T1} - d$. The above synchronization scheme is referred to as the H-sensor broadcast synchronization (HBS) scheme.

Although the HBS scheme cannot completely prevent the *DoS attack*, it makes the detection of the *DoS attack* much easier than in a homogeneous sensor network. In an HSN, only a small number of H-sensors serve as the reference nodes for time synchronization. When an HSN broadcasts a timing message, other nodes are not supposed to transmit any packet. If a node always transmits packets that overlap with timing messages, the sender is probably an attacker. When a neighbor L-sensor (or H-sensor) observes the event, it can generate an alarm, and then, various techniques can be used to defend the *DoS attack.* The defense against the *DoS attack* is out of the scope of this paper. In the rest of this paper, we will focus on the defense against the four attacks mentioned in Section I.

To achieve secure time synchronization in an HSN, an authentication scheme is required to ensure that a time reference broadcast message is from an actual cluster head because an adversary can launch the following attack. The adversary can deploy a powerful node D in the network. Node D broadcasts false time reference messages to L-sensors and, thus, disrupts time synchronization in the network. We propose two approaches to authenticate timing messages from H-sensors.

C. Authenticating Broadcasts by Neighbor H-Sensors

In the first approach, broadcast messages from an H-sensor are authenticated by neighbor H-sensors. Because H-sensors are powerful high-end sensors, it is reasonable to assume that H-sensors are equipped with tamper-resistant hardware. The number of H-sensors in an HSN is relatively small, and thus, the total cost of tamper-resistant hardware is small. Assume that H-sensors can run public-key algorithms such as RSA because they have strong computation capability. Furthermore,

the recent progress in elliptic curve cryptography (ECC) [17] provides new opportunities to utilize public-key cryptography, even for low-end sensors. The recent implementation [18] of a 160-bit ECC on Atmel ATmega128, with a central processing unit of 8 Hz and 8 bits, shows that an ECC point multiplication takes less than 1 s. The above fact demonstrates that the ECC public-key cryptography is feasible for small low-end sensors. Assume that that ECC is chosen as the public key algorithm.

Suppose that there is a total of M H-sensors in the network. Before sensor deployment, M pairs of ECC public/private keys are generated and each pair is assigned an index. Each H-sensor is preloaded with one unique private key and all of the M public keys. The preloaded public/private keys are protected by the tamper-resistant hardware in the H-sensor. Even when an H-sensor is captured, the keys are not revealed. After sensor deployment and cluster formation, each H-sensor broadcasts a timing message to all L-sensors in its cluster. A timing broadcast from one H-sensor can reach the neighbor H-sensors. The timing message includes the following fields: Timestamp + Sequence number + Key index + MAC, where Timestamp is the local time read from the H-sensor's clock, Key index indicates which private key is used for the MAC calculation, and MAC is computed over all the previous fields by using the private key. Neighbor H-sensors have the corresponding public key, and they can check if the MAC is correct and authenticate the timing message, i.e., if the message is from a valid H-sensor. If the timing message fails to pass the authentication check, an alarm message will be broadcasted by the neighbor H-sensor to all L-sensors in that cluster.

The above H-sensor-based authentication scheme can defend against all of the four attacks mentioned in Section I. Because only a legitimate H-sensor knows the private key, an attacker is not able to pretend to be a cluster head and broadcast timing messages with a valid MAC. This prevents the *masquerade attack*. Each timing message includes a message sequence number, and this prevents the *replay attack*. The broadcast from an H-sensor can directly reach all L-sensors in the cluster. An attacker does not have the chance to relay the timing message; thus, it cannot drop or modify the message. The one-hop broadcast from H-sensors prevents the *message manipulation attack*. The one-hop broadcast also prevents the *delay attack* for the same reason.

The above security analysis shows that the H-sensor-based authentication scheme is very effective in preventing various attacks on time synchronization. However, because H-sensors are randomly distributed in the network, it is possible that an H-sensor does not have any neighbor H-sensor within the transmission range. In such a case, the above authentication scheme does not work. To solve the problem, we propose an effective scheme where L-sensors authenticate the broadcast timing message as follows.

D. Authenticating Broadcasts by L-Sensors

In this section, we present a distributed message authentication scheme in which L-sensors authenticate timing messages from the H-sensor. We refer to this scheme as the subset authentication (SA) scheme. Based on the key management scheme [19] for HSNs, each L-sensor has a unique shared key with the cluster head, and the cluster head knows all the shared keys. Each time a cluster head H wants to broadcast a timing message, it selects k keys from the keys shared between H- and L-sensors in its cluster, where k is a system parameter. The keys are selected by following a predefined pattern, e.g., one key from each L-sensor in a round-robin way, and L-sensors are ordered according to their node IDs. During the key setup phase, the cluster head learns the node IDs of all L-sensors (in the cluster) and all the IDs of the preloaded keys in the L-sensors. The cluster head broadcasts the list of L-sensor IDs and key IDs to the entire cluster, and then, each L-sensor records the IDs of the L-sensors and keys in the cluster (note that the key IDs are used for other defense, as discussed below) and, hence, knows the node-selection pattern. For example, suppose that there are 30 L-sensors in a cluster and that k is 8. For simplicity, assume that the node IDs of the L-sensors are 1, 2, ..., 30. The SA scheme selects one key from each of eight L-sensors for every broadcast. For example, L-sensors 1–8 are selected for the first broadcast, L-sensors 9–16 are selected for the second broadcast, L-sensors 17–24 are selected for the third broadcast, L-sensors 25–30, 1, and 2 are selected for the fourth broadcast, and so on. The node-selection pattern needs to be known by all L-sensors to avoid the following attack. If a powerful attacker D has compromised a few L-sensors, D can select the keys only from the compromised L-sensors, then D's broadcast messages can pass the authentication check. Note that if the node subset is randomly selected, then the broadcast authentication scheme is vulnerable to the above attack.

When a cluster head H wants to broadcast a timing message, H selects k keys and computes k MACs using the keys. The MACs and the IDs of the keys used to calculated the MACs are included in the broadcast message, i.e., message$\|$(key_ID1 + MAC1)$\|$(key_ID2 + MAC2) ... $\|$(key_IDk + MACk), where MACi is calculated from the "message" using key_IDi, and $i = 1, 2, \ldots, k$. If an L-sensor (denoted as u) owns one of the keys, u can check if the corresponding MAC is correct and, hence, authenticate the message. If the MAC cannot pass the check, the L-sensor will flood an alarm message to other L-sensors in the cluster to notify them that the previous broadcast is not from a valid cluster head. Note that a broadcast message is authenticated by at least k L-sensors because a key may be preloaded in more than one L-sensor.

An attacker can launch the following attack. A powerful attacker D broadcasts a timing message with all the MACs computed by keys unknown to the L-sensors in the cluster. To defend against this attack, we require that every key used for MAC calculation should be owned by at least one L-sensor in the cluster. Because every L-sensor knows the IDs of all the keys owned by the L-sensors from the H-sensor broadcast, it is straightforward to check if a key is owned by any L-sensor in the cluster.

With the above authentication scheme, an adversary needs to compromise at least k L-sensors before it can forge a broadcast message from the cluster head. Thus, the above scheme provides k-resistance against the node compromise attack. Furthermore, the keys used to calculate the MACs are different for different broadcasts. Hence, it is very hard for an adversary

to forge multiple broadcast messages unless it compromises a large number of L-sensors in one cluster. In case an adversary has compromised most L-sensors, no security scheme would work well. The k number of MACs used for authentication is an important parameter. The larger the value of k, the more resilience against the node compromise attack and hence better security. On the other hand, a larger value of k means more overhead of the authentication scheme. The optimal value of k should depend on the cluster size (e.g., the number of L-sensors), the node capability of L- and H-sensors, etc. It is our future work to determine the optimal value of k such that security and performance are well balanced. To increase the resilience against the L-sensor compromise attack, a cluster head may periodically broadcast time reference messages. Because the keys used to compute the MACs are different for different broadcasts, an attacker would not be able to forge multiple timing messages if only part of the L-sensors is compromised.

The SA scheme can defend against the four attacks on time synchronization. A broadcast message is authenticated by at least k L-sensors, and an adversary is not able to masquerade a cluster head, unless a large number of L-sensors are compromised. The keys used to calculate the MACs change for different broadcast messages, and thus, a replay of an old timing message can be detected. In addition, sequence number may be included in the timing message to defend the *replay attack*. Because the broadcast message from a cluster head reaches every L-sensor in the cluster by one hop, an attacker would not be able to launch the *message manipulation attack* (i.e., to drop or modify a timing message). For the same reason, an attacker would not be able to launch the *delay attack* (i.e., to deliberately delay timing messages).

IV. Performance Evaluations

In this section, we evaluate the performance of our time synchronization for HSNs. Specifically, we evaluate the synchronization performance using real sensors in Section IV-A, analyze the security performance in Section IV-B, and show the significant saving on communication overhead in Section IV-C.

A. Synchronization Performance

We compare the synchronization performance of HBS and TPSN via experiments using real sensors. The HBS scheme is implemented in Tmote Invent sensors [20]. For comparison, the TPSN scheme is also implemented in Tmote sensors. In the first set of experiments, two Tmote sensors (H and L) are used for performance comparison. The clocks at the two Tmotes were randomly started. One Tmote H serves as the H-sensor and sends out timing messages. The other Tmote L serves as the L-sensor and receives timing messages. The distance between H and L is known to node L so that L can estimate the propagation delay. After sending messages, the synchronization error is calculated by observing the phase shift between the two clock waveforms (of sensor H and L) on a digital analyzer. The experiments run 100 times. For the same setting, TPSN is run by the two sensors H and L, where L sends out the first message, and H sends out the second message. Then, sensor L can estimate the clock drift between L and H, based on T1, T2, T3, and T4. We run the experiments for 100 times. As discussed earlier, the clock drift in HBS is $\Delta = \mathrm{T2} - \mathrm{T1} - d$, and the clock drift in TPSN is $\Delta = [(\mathrm{T2} - \mathrm{T1}) - (\mathrm{T4} - \mathrm{T3})]/2$. The statistics of the synchronization errors are shown in Table I.

TABLE I
Statistics of Synchronization Errors

	HBS	TPSN
Average error (μs)	10.6	14.2
Worst case error (μs)	36.5	39.0
Best case error (μs)	0.0	0.0

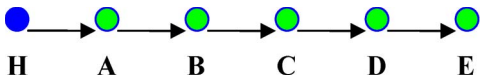

Fig. 2. Multihop experiments.

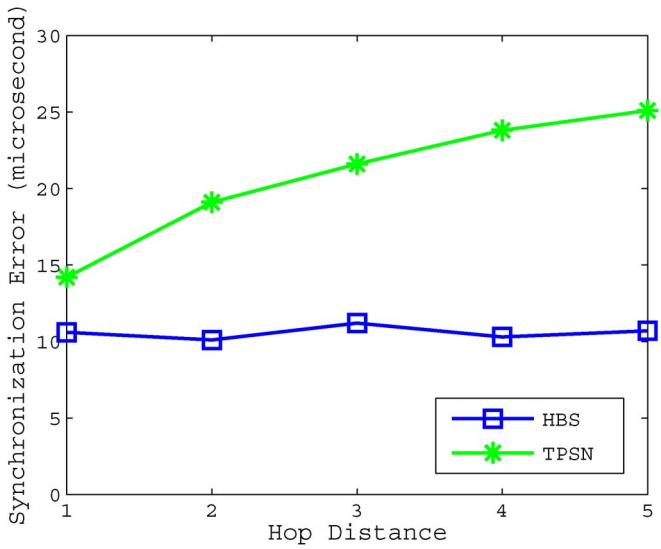

Fig. 3. Average synchronization error for multihop communications.

In the second set of experiments, we compare the performance of HBS and TPSN using multihop communications. In an outdoor environment, a total of six Tmote sensors are laid out in a line, as illustrated in Fig. 2. The distance between two neighbor sensors is 10 ft. For the HBS scheme, sensor H serves as the H-sensor, and the other five sensors serve as L-sensors. In each experiment, H broadcasts a timing message, and all the other five sensors receive the message. Each sensor knows the locations of H and itself. For the TPSN scheme, a multihop synchronization scenario is simulated, where A synchronizes its clock with H, B synchronizes its clock with A, ..., and E synchronizes its clock with D. Using the same approach as in the first set of experiments, the synchronization error is calculated by observing the phase shift between the clock waveforms on a digital analyzer.

The average synchronization errors with multihop communications are plotted in Fig. 3. This figure shows that the average synchronization error under TPSN is always larger than that under HBS for different hop distances. In addition, Fig. 3 shows that the average synchronization error under TPSN increases

as the hop distance increases, whereas the average synchronization error under HBS is almost the same for different hop distances. For TPSN, a certain amount of synchronization error is introduced for each hop, and the error accumulates as hop distance increases. However, in HBS, the H-sensor only needs one broadcast to synchronize all L-sensors, i.e., it is still single-hop in HBS. There is no accumulation of synchronization errors in HBS, and thus, the errors for different hops are about the same. This experiment demonstrates an advantage of using HBS, i.e., because of the long transmission range of H-sensors, the synchronization error under HBS is not accumulated for multihop L-sensors. Note that the hop here is L-sensor's hop.

B. Security Performance of the L-Sensor-Based Authentication

The discussion in Section III shows that our time synchronization scheme for HSN can defend against the four attacks (mentioned in Section I) on synchronization. In this section, we analyze the security performance of the authentication scheme based on L-sensors, i.e., the resilience against the node compromise attack. We compute the probability that an attack could generate a valid timing broadcast message when different numbers of L-sensors are compromised. Assume that the total number of L-sensors in a cluster is N and C L-sensors are compromised. We consider two cases. In the first case, the C L-sensors are randomly selected from the N L-sensors. This case corresponds to the scenario where the attacker cannot select the sensors that it wants to compromise, e.g., the attacker only has access to certain areas of the cluster. The first case is referred to as random compromise (RC) attack. In the second case, the attacker can select whichever L-sensors to compromise. The second case is referred to as selected compromise (SC) attack. The SC attack can do more damage to the network than the RC attack.

For the RC attack, we calculate the probability that an attacker knows all the k keys (used to generate the k MACs) after compromising C out of N L-sensors. Below, we refer to this probability as attack probability. In RC attack, the L-sensors are randomly selected. The number of different combinations of choosing C out of N is $\binom{N}{C}$. The number of combinations of choosing C out of N and including k particular keys in the C keys is $\binom{N-k}{C-k}$. Thus, the attack probability is $\binom{N-k}{C-k}/\binom{N}{C}$.

For the SC attack, the attacker can choose any sensor to compromise. If the attacker knows the key selection pattern, then it can forge a timing broadcast message by compromising only k keys. However, the attacker can only forge one timing message if it only compromises k keys because the keys used for different timing messages are different. In Fig. 4, we plot the attack probability under RC and SC attacks. The total number of L-sensors in a cluster (N) is 100 and 200 in Fig. 4(a) and (b), respectively. The number of compromised L-sensors is increased from 1 to N. We compare the attack probability under the RC and SC attacks for different values of k, which is the number of MACs used to protect one broadcast message.

Fig. 4 shows that the attack probability under the RC attack is very small (close to zero), even when 60% of the L-sensors are compromised, i.e., an attacker is not able to forge a timing broadcast message, unless it compromises a large portion of the L-sensors. Fig. 4 shows that even when 70% or 80% of the L-sensors are compromised, the attack probability is still low, e.g., when $N = 200$ and 160 L-sensors (80%) are compromised, the attack probability is only 0.1013, 0.0089, and 0.0007 for $k = 1, 2, 3$, respectively. We plot the probability curves for k, being 10, 20, and 30. Fig. 4 also shows that the attack probability is significantly reduced as k increases. The reason is straightforward—the more keys used to protect a broadcast message, the harder an attacker forges the message.

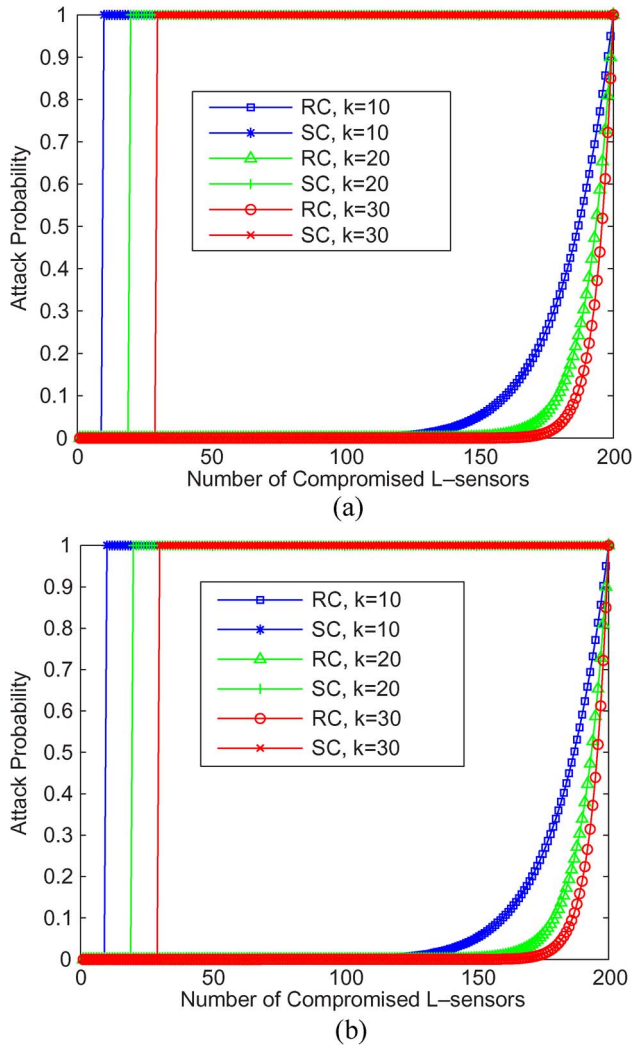

Fig. 4. Attack probability. (a) $N = 100$. (b) $N = 200$.

Under the SC attack, an attacker can choose any L-sensor to compromise (this may be a strong assumption for the attacker). Thus, whenever the number of compromised node is larger than k, the attacker can forge one broadcast message. However, because different keys are used to protect different broadcast messages, an attacker is only able to forge one broadcast message when it breaks k L-sensors. Furthermore, the usage of keys follows a certain pattern, e.g., in the round-robin way according to the key index, and thus, the attacker cannot launch the attack at any time. In other words, it can only fake a broadcast message when it knows all of the keys being used. This limits the damage of the attack. Furthermore, as long as the cluster head H is available, H can still broadcast

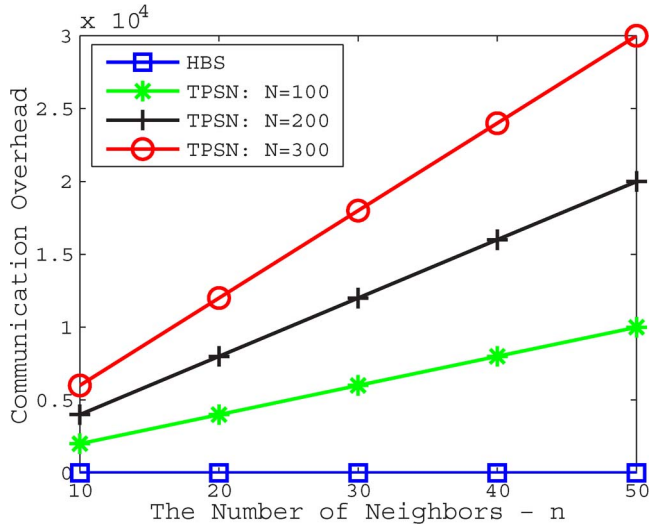

Fig. 5. Communication overhead for synchronization.

its message, which is protected by these keys. An L-sensor can send out alarm messages when it receives two broadcast messages protected by the same subset of keys during a short time period, and then, the attack is detected.

To sum up, our security analysis shows that the L-sensor-based authentication scheme is resilient to the node compromise attack.

C. Significant Savings on Communication Overhead

Another great advantage of the HBS scheme is the small communication overhead. In HBS, only one broadcast from the H-sensor can synchronize all L-sensors in a cluster. Assume that there are a total of N L-sensors in a cluster and that the average number of neighbors of an L-sensor is n. Using the HBS scheme, the number of transmission is one. Using TPSN, two transmissions are required for an L-sensor to synchronize its clock with one of its neighbors (as shown in Fig. 1). If there are n neighboring L-sensors, the number of timing messages exchanged is $2n$. For a cluster with N L-sensors, the total number of timing messages for synchronization is $2nN$. In Fig. 5, we plot the number of transmissions for different values of n and N, where the y-axis is the number of timing messages. As we can see in Fig. 5, the more L-sensors there are in one cluster (a larger N) and the denser the L-sensors (a larger n) are, the more transmissions TPSN will require, and the more savings on communication overhead HBS will achieve.

V. Conclusion

In this paper, we have presented a secure efficient accurate time synchronization scheme for HSNs. The scheme consists of the following two parts: 1) synchronizing all H-sensors in an HSN and 2) synchronizing L-sensors in each cluster. We utilize the long transmission range of H-sensors, i.e., one broadcast from an H-sensor can synchronize all L-sensors in a cluster. The one-hop broadcast makes the scheme resilient to several attacks on time synchronization because an attacker does not have the chance to relay (hence manipulate) the timing messages. One limitation of our synchronization scheme is that it cannot completely prevent the *DoS attack*. In general, it is not easy to defend the *DoS attack* in wireless sensor networks. However, our scheme does make the detection of the *DoS attack* much easier than in a homogeneous sensor network. Our experiments on Tmote sensors show that the time synchronization scheme has better accuracy than TPSN. Our security analysis demonstrates that the time synchronization scheme can defend against the four attacks on synchronization, and it is resilient to node compromise attack. The performance analysis shows that our scheme achieves significant savings on communication overhead.

References

[1] D. Tian and N. D. Georganas, "A coverage-preserved node scheduling scheme for large wireless sensor networks," in *Proc. 1st Int. Workshop WSNA*, Atlanta, GA, Sep. 2002, pp. 32–41.

[2] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Wireless Ad Hoc Sensor Netw.: Int. J.*, vol. 1, no. 1/2, pp. 89–123, Oct. 2004.

[3] W. Zhang and G. Cao, "Optimizing tree reconfiguration for mobile target tracking in sensor networks," in *Proc. IEEE INFOCOM*, 2004, pp. 2434–2445.

[4] W. Zhang and G. Cao, "DCTC: Dynamic convoy tree-based collaboration for target tracking in sensor network," *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, pp. 1689–1701, Sep. 2004.

[5] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. ICDCS*, Vienna, Austria, Jul. 2002, p. 457.

[6] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. NDSS*, Feb. 2001, pp. 35–46.

[7] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. WCNC*, 2003, pp. 1266–1273.

[8] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st Int. Conf. Embedded Netw. Sensor Syst.*, 2003, pp. 138–149.

[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2nd ACM SenSys*, Baltimore, MD, Nov. 2004, pp. 39–49.

[10] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," in *Proc. 2nd IEEE Int. Conf. MASS*, Washington, DC, Nov. 2005, pp. 765–772.

[11] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. 6th ACM Mobicom*, 2000, pp. 255–265.

[12] K. Sun, P. Ning, and C. Wang, "Fault-tolerant cluster-wise clock synchronization for wireless sensor networks," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 3, pp. 177–189, Jul.–Sep. 2005.

[13] L. Girod *et al.*, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *Proc. ACM SenSys*, 2004, pp. 201–213.

[14] M. Yarvis, N. Kushalnagar, H. Singh *et al.*, "Exploiting heterogeneity in sensor networks," in *Proc. IEEE INFOCOM*, Miami, FL. Mar. 2005, pp. 878–890.

[15] X. Du and F. Lin, "Maintaining differentiated coverage in heterogeneous sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 5, no. 4, pp. 565–572, Sep. 2005.

[16] L. Lazos and R. Poovendran, "SeRLoc: Secure range-independent localization for wireless sensor networks," in *Proc. ACM WiSe*, Philadelphia, PA, 2004, pp. 21–30.

[17] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, Jan. 1987.

[18] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Proc. 6th Int. Workshop Cryptographic Hardware Embedded Syst.*, Boston, MA, Aug. 2004, pp. 119–132.

[19] X. Du, Y. Xiao, M. Guizani, and H. H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Netw.*, vol. 5, no. 1, pp. 24–34, Jan. 2007.

[20] *Moteiv Sensor Nodes*. [Online]. Available: http://www.moteiv.com
[21] F. Hu, Y. Malkawi, S. Kumar, and Y. Xiao, "Vertical and horizontal synchronization services with outlier detection in underwater acoustic networks," *Wireless Commun. Mobile Comput.,* to be published.

Xiaojiang Du (M'03) received the B.E. degree from Tsinghua University, Beijing, China, in 1996 and the M.S. and Ph.D. degrees from the University of Maryland, College Park, in 2002 and 2003, respectively, all in electrical engineering.

He is currently an Assistant Professor with the Department of Computer Science, North Dakota State University, Fargo. He is an Associate Editor for four international journals. His research interests include heterogeneous wireless sensor networks, security, wireless networks, computer networks, network and systems management, and controls. He is the author of over 50 published journal and conference papers in the above areas.

Mohsen Guizani (SM'99) received the B.S. (with distinction) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, in 1984, 1986, 1987, and 1990, respectively.

He is currently a Full Professor and the Chair of the Computer Science Department, Western Michigan University, Kalamazoo. He is the founder and the Editor-In-Chief of *Wiley Wireless Communications and Mobile Computing Journal*. His research interests include computer networks, design and analysis of computer systems, wireless communications, and optical networking. He has authored or coauthored over 180 technical papers published in major international journals and conference proceedings. He currently serves on the editorial boards of many national and international journals.

Yang Xiao (SM'04) received the B.S. and M.S. degrees from Jilin University, Jilin, China, in 1989 and 1991, respectively, and the M.S. and Ph.D. degrees in computer science and engineering from Wright State University, Dayton, OH, in 2000 and 2001, respectively.

He is currently with the Department of Computer Science, University of Alabama, Tuscaloosa, and currently serves as the Editor-in-Chief of the *International Journal of Security and Networks*, the *International Journal of Sensor Networks*, and the *International Journal of Telemedicine and Applications*. His research interests include security, telemedicine, sensor networks, and wireless networks. He is the author of more than 200 papers published in major journals and refereed conference proceedings, as well as book chapters related to these research areas.

Dr. Xiao serves as an Associate Editor for several journals, including the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Hsiao-Hwa Chen (SM'00) received the B.Sc. and M.Sc. degrees from Zhejiang University, Hangzhou, China, in 1982 and 1985, respectively, and the Ph.D. degree from the University of Oulu, Oulu, Finland, in 1990, all in electrical engineering.

He is currently a Full Professor with the Department of Engineering Science, National Cheng Kung University, Taipei, Taiwan, R.O.C. He is also an Adjunct Professor with Zhejiang University and Shanghai Jiao Tong University, Shanghai, China. He has authored or coauthored over 200 technical papers published in major international journals and conferences, as well as six books in the areas of communications. He has served or is serving as an Editor and/or Guest Editor for many international journals.

Dr. Chen has served as a Symposium Cochair of major international conferences, including the IEEE Vehicular Technology Conference, the IEEE International Conference on Communications, the IEEE Global Telecommunications Conference, and the IEEE Wireless Communication Networking Conference.