Secure Time Synchronization in Wireless Sensor Networks: A Maximum Consensus-Based Approach

Jianping He, Jiming Chen, Peng Cheng, and Xianghui Cao

Abstract—Time synchronization is a fundamental requirement for the wide spectrum of applications with wireless sensor networks (WSNs). However, most existing time synchronization protocols are likely to deteriorate or even to be destroyed when the WSNs are attacked by malicious intruders. This paper is concerned with secure time synchronization for WSNs under message manipulation attacks. Specifically, the theoretical analysis and simulation results are first provided to demonstrate that the maximum consensus based time synchronization (MTS) protocol would be invalid under message manipulation attacks. Then, a novel secured maximum consensus based time synchronization (SMTS) protocol is proposed to detect and invalidate message manipulation attacks. Furthermore, we prove that SMTS is guaranteed to converge with simultaneous compensation of both clock skew and offset. Extensive numerical results show the effectiveness of our proposed protocol.

Index Terms—Wireless sensor networks, time synchronization, cyber physical security, maximum consensus

1 Introduction

Time synchronization is crucial for many applications, e.g., event detection, speed estimating, environment monitoring, etc., in wireless sensor networks (WSNs), as these applications need that all sensor nodes have a common time reference [1]. Moveover, the time synchronization also provides the possibility to schedule the sensor activation for energy conservation [2]. Different protocols have been developed for time synchronization of WSNs in various scenarios, e.g., [3], [4], [5], [6], [7], [8], [9].

However, until recently, cyber physical security in WSNs is becoming a hot while challenging research area [10], [11], [12]. Due to the small-size as well as low-cost requirement of sensor nodes, the present hardware and software in WSNs are quite vulnerable to various malicious cyber physical attacks. There are mainly two kinds of attacks [19], [21]. One is to attack the sensor nodes directly. Due to limited resources, current battery powered sensor nodes are prone to various failure and malfunctions. Besides, they may also be easily compromised by adversaries as to generate false messages [22]. The other is to attack the communication links as the wireless media are shared among nodes at the dedicated frequencies, which are vulnerable to attacks such as congestion, eavesdropping, falsification, and injection. Thus, it is significant and challenging to guarantee the WSNs performance against such cyber physical attacks [14], [24].

• *The authors are with the State Key Laboratory of Industrial Control Technology, Department of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China.*
E-mail: {jphe, jmchen, pcheng, xhcao}@iipc.zju.edu.cn.

Manuscript received 26 Sept. 2012; revised 5 Mar. 2013; accepted 13 May 2013; date of publication 26 May 2013; date of current version 21 Feb. 2014.
Recommended for acceptance by C. Pinotti.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.150

Especially, cyber physical attacks to network time synchronization may incur data disordering, unsynchronized task execution and duty-cycling, and even malfunctions, which will degrade the whole network performance. For instance, in the IEEE 802.15.4 standard, its medium access control (MAC) protocol often requires sensor nodes to maintain a common time frame as well as a common unit time slot. Attacks that break such synchronization may increase interferences, packet collisions, and communication delay. Therefore, secure time synchronization becomes a critical requirement for WSN in order to provide the secured system services.

Consensus based time synchronization protocols are developed to overcome the shortages of traditional time synchronization protocols in terms of increasing the robustness and accuracy of synchronization [6], [7], [8]. Unlike the traditional time synchronization protocols, consensus based time synchronization protocols remove the tree topology requirement and do not rely on any specific sensor node as they are completely distributed. Meanwhile, as pointed in [6], the consensus based protocols can obtain more accurate synchronized clocks between neighbor nodes than traditional tree-based protocols. Specifically, Schenato and Fiorentin [7] propose an average time-sync (ATS) protocol, which consists of two averaging consensus algorithms. Nevertheless, it generally requires a large amount of data exchanges and the converging speed may be quite slow when the network size grows large. To this end, a maximum consensus based time synchronization protocol (MTS) is proposed in [8]. It has been shown that MTS converges faster than ATS. Meanwhile, these consensus based protocols are able to compensate both clock skew and offset simultaneously. Unfortunately, since neither ATS nor MTS has considered the security problem under cyber physical attacks, both of them will fail to synchronize under message manipulation attacks.

In this paper, we would like to develop a secured and distributed time synchronization protocol, which is able to achieve fast time synchronization even under the message manipulation attacks. To the best of our knowledge, there is no distributed secure time synchronization protocol, which is able to compensate both clock skew and offset simultaneously. The main contributions of this paper are summarized as follows:

1. The problem of secure time synchronization for WSNs under *Message manipulation* attacks is formulated, where both clock skews and offsets are required to be synchronized.
2. Both theoretical and simulation analysis are provided to illustrate that the existing MTS would be invalid under message manipulation attacks. We discuss the main challenges and opportunities for consensus based time synchronization under message manipulations.
3. Based on the maximum consensus concept, a novel secure time synchronization protocol, secured maximum consensus based time synchronization (SMTS), is proposed. We propose both hardware clock and logical clock checking processes as the safeguard mechanisms, so that it can detect and invalidate the possible message manipulation attacks. Meanwhile, we provide the performance analysis of SMTS in terms of energy cost. Simulated results are conducted to evaluate the effectiveness of SMTS.

The remainder of this paper is organized as follows. In Section 2, the problem of secure time synchronization is formulated. Section 3 analyzes the performance of MTS under message manipulation attacks. We propose the SMTS protocol and prove its convergence in Section 4. Simulation results are presented in Section 5. Section 6 provides the related works. Finally, Section 7 concludes this paper.

2 Problem Formulation

Consider a sensor network with n safe nodes and m malicious nodes (attack nodes), where the attack nodes can be external attackers or in-network nodes compromised by attackers and assume $m < n$. Assume that these $n+m$ nodes have different and unique identity numbers, e.g., the safe and attack nodes are respectively indexed by $1, 2, \ldots, n$ and $n+1, n+2, \ldots, n+m$. Each node will only know whether itself is an attack node without any advanced knowledge about other nodes. Let $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$ denote the graph of whole network, where $\mathcal{V}$ denotes the set of vertexes (nodes) and $\mathcal{E}(t)$ is the set of communication links. Similarly, let $\mathcal{G}_a(t) = (\mathcal{V}_a, \mathcal{E}_a(t))$ and $\mathcal{G}_s(t) = (\mathcal{V}_s, \mathcal{E}_s(t))$ denote the graphs composed by the attack nodes and by the safe nodes, respectively. It is straightforward that $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_a$. We assume that $\mathcal{G}_s(t)$ is connected,[1] which is a basic assumption for distributed clock synchronization protocols [26]. In this paper, we focus on the situation where the communication delay is ignorable compared with the broadcasting periods of different nodes. Assume that each node can set the authenticated message, such that it can be seen but cannot be modified by other nodes, which is also used in [19], [25], [27], where a message authentication code is applied. Table 1 gives some important notations.

1. Note that this assumption can be relaxed as that $\mathcal{G}_s(t)$ is joint connected, which is the same as Assumption 1 [8].

TABLE 1
Notation Definitions

Symbol	Definition
$\tau_i(t)$	the hardware clock reading of node i at time t;
$L_i(t)$	the logical clock reading of node i at time t;
$\hat{a}_i$	the adjusting parameter for logical clock skew compensation;
$\hat{b}_i$	the adjusting parameter for logical clock offset compensation;
a_{ij}	the relative clock skew;
q_{ij}	the ratio of logical clock skews;
x_i	the logical clock skew of node i;
$\mathcal{N}_i$	the neighbor node set of node i;
$\{\cdot\}_j^e$	the information of an attack node j;
$\mathbf{N}^+$	the set of positive integers;
t^+	the time just after updating at time t.

2.1 Clock Model

It is widely adopted that the hardware clock reading $\tau_i(t)$ of any node $i \in \mathcal{V}$ at time t can be modeled as the following linear function, [6], [7], [8],

$$\tau_i(t) = a_i t + b_i, \quad i \in \mathcal{V}, \tag{1}$$

where a_i is the hardware clock skew which determines the clock speed and b_i is the hardware clock offset. In the ideal case, $a_i = 1$ and $b_i = 0$. However, practical clocks have different skews and offsets in general, i.e., $a_i \neq a_j, i \neq j$.

It has been pointed out that a_i and b_i cannot be exactly calculated [7]. However, by comparing the local clock readings, the hardware clock of node i can also be expressed as follows:

$$\tau_i(t) = \frac{a_i}{a_j}\tau_j(t) + \left(b_i - \frac{a_i}{a_j}b_j\right) = a_{ji}\tau_j(t) + b_{ji}, \tag{2}$$

where $a_{ji} = \frac{a_i}{a_j}$ is the relative hardware clock skew [8], and $b_{ji} = b_i - a_{ji}b_j$ is the relative hardware clock offset, both of which can be estimated based on the hardware readings of node i and j [7].

The relative skew a_{ij} is defined as $a_{ij} = \frac{a_j}{a_i}$, which is estimated by

$$a_{ij}(t_1) = \frac{\tau_j(t_1) - \tau_j(t_0)}{\tau_i(t_1) - \tau_i(t_0)}, \quad i, j \in \mathcal{V}, \tag{3}$$

where $(\tau_i(t_1), \tau_j(t_1))$ and $(\tau_i(t_0), \tau_j(t_0))$ are the hardware clock readings of node i and j at time instances t_1 and t_0, with $t_1 > t_0$. In detail, once node i receives time message $\tau_j(t_0)$ from node j, it reads its current clock and temporally stores $(\tau_i(t_0), \tau_j(t_0))$. Clearly if node i receives the time message $\tau_j(t_1)$ from node j for the second time, the relative skew a_{ij} can be obtained by (3) directly. After obtaining relative skew a_{ji}, the relative hardware clock offset b_{ji} can be obtained from (2) immediately, i.e., $b_{ji} = \tau_i(t) - a_{ji}\tau_j(t)$.

Since manually adjusting the hardware clock skew or offset is nearly infeasible [6], we can define a logical clock $L_i(t)$ to replace the hardware clock as follows:

$$L_i(t) = \hat{a}_i(t)\tau_i(t) + \hat{b}_i(t) = \hat{a}_i(t)a_i t + \hat{a}_i(t)b_i + \hat{b}_i(t),$$

where $\hat{a}_i(t)$ and $\hat{b}_i(t)$ are two adjusting parameters, which are used for time synchronization.

2.2 Attack Model

Time synchronization protocols in WSNs are vulnerable to a number of security attacks including, sybil attack, replay attack, message manipulation attack, delay attack and Dos attack, etc., [16], [19]. In this paper, we mainly consider the attack nodes which do not know the identity of each other and cannot collude. We will only focus on the message manipulation attack mode, which is defined as follows.

Message manipulation includes dropping and transmitting fake synchronization messages. For instance, an attacker pretends as a safe node and corrupts the synchronization information, e.g., hardware clock reading and adjusting parameters, and broadcasts to its neighbor nodes. In this way, the attack nodes can mislead their neighbor nodes and damage the synchronization [14], [25].

From the definition of Message manipulation, it follows that the replay attack, delay attack and fault data injection attack can also be viewed as the different kinds of message manipulation. For example, replay attack can be modelled as adding a negative time to the real message, while delay attack can be viewed as adding a delay to the real message. Since we focus on the maximum consensus based time synchronization, the information for nodes communication includes hardware clock readings and adjusting parameters. Thus, we assume that the attackers has the ability to freely manipulate and broadcast the fake hardware clock readings and adjusting parameters if they decide to attack.

2.3 Problem Setup

For each hardware clock $\tau_i(t)$, there always exists a pair of $(\hat{a}_i, \hat{b}_i)$, such that

$$L_i(t) = \hat{a}_i\tau_i(t) + \hat{b}_i = \tau_v(t),\ i \in \mathcal{V},$$

where $\tau_v(t) = a_v t + b_v$ is a common clock, and where a_v and b_v are two constants.

Hence, the goal of traditional time synchronization protocol is to find $(\hat{a}_i, \hat{b}_i)$ for $\forall i \in \mathcal{V}$, such that all nodes' logical clocks are equal to the common clock $\tau_v(t)$, and hence achieve synchronization. However, in this paper, aside from that all the safe nodes still aim to synchronize their logical clocks, the attack nodes aim to degrade the time synchronization as much as possible. Therefore, our goal is to design a clock synchronization protocol to find a pair of $(\hat{a}_i(k), \hat{b}_i(k))$ for each safe node $i \in \mathcal{V}_s$, such that

$$\begin{cases} \lim\limits_{k\to\infty} \hat{a}_i(k)a_i = a_v, \\ \lim\limits_{k\to\infty} \hat{a}_i(k)b_i + \hat{b}_i(k) = b_v, \end{cases}$$

where k is the iteration of the protocol.

3 MTS under Attacks

3.1 MTS Protocol and Message Manipulation

The skew and offset compensation strategies of MTS are described as follows:

$$\hat{a}_i(t^+) = \max\{\hat{a}_i(t), a_{ij}(t)\hat{a}_j(t)\}, \tag{4}$$

$$\hat{b}_i(t^+) = \begin{cases} L_j(t) - \hat{a}_i(t^+)\tau_i(t), & q_{ij}(t) > 1, \\ L_{max}^{ij}(t) - \hat{a}_i(t)\tau_i(t), & q_{ij}(t) = 1, \\ \hat{b}_i(t), & q_{ij}(t) < 1, \end{cases} \tag{5}$$

where $j \in \mathcal{N}_i$, $L_{max}^{ij}(t) = \max\{L_i(t), L_j(t)\}$, and $q_{ij}(t)$ is the ratio of logical clock skews, computed by $q_{ij}(t) = \frac{a_{ij}(t)\hat{a}_j(t)}{\hat{a}_i(t)} = \frac{a_j(t)\hat{a}_j(t)}{a_i(t)\hat{a}_i(t)}$. From (4) and (5), it can be observed that node i will select its neighbor node j as the reference node when node j has larger logical clock skew or has the same logical clock skew but larger logical clock.

The attacker may manipulate the message in a random way to destroy the time synchronization. For example, let node j be the attack node, which broadcasts fake messages with hardware clock reading $\tau_j^e(t_k)$ and logical clock adjusting parameters $\hat{a}_j^e(t_k)$ and $\hat{b}_j^e(t_k)$, where the values of these fake messages can be arbitrarily chosen by node j at each time $t_k, k \in \mathbf{N}^+$. Thus, when a safe node i receives such fake message τ_j^e from node j, it will estimate the relative skew according to

$$\begin{aligned} a_{ij}^e(t_1) &= \frac{\tau_j^e(t_1) - \tau_j^e(t_0)}{\tau_i(t_1) - \tau_i(t_0)} \\ &= \frac{\tau_j^e(t_1) - \tau_j^e(t_0)}{\tau_j(t_1) - \tau_j(t_0)} \frac{\tau_j(t_1) - \tau_j(t_0)}{\tau_i(t_1) - \tau_i(t_0)} \\ &= \delta_j^e(t_1)a_{ij}, \end{aligned}$$

where $\delta_j^e(t_1) = \frac{\tau_j(t_1^e) - \tau_j(t_0^e)}{\tau_j(t_1) - \tau_j(t_0)}$ is the value of the fake hardware clock distance over the true distance between two consecutive communication times. Since the node j is able to change the value of $\tau_j^e(t)$ freely, it can determine $\delta_j^e(t)$. Hence, based on fake messages τ_j^e and $\hat{a}_j^e$, the skew compensation will be rewritten as

$$\begin{aligned} \hat{a}_i(t^+) &= \max\{\hat{a}_i(t), a_{ij}^e(t)\hat{a}_j^e(t)\} \\ &= \max\{\hat{a}_i(t), a_{ij}\delta_j^e(t)\hat{a}_j^e(t)\}. \end{aligned} \tag{6}$$

From (6), it follows that both fake messages τ_j^e and $\hat{a}_j^e$ can directly affect the skew compensation as well as offset compensation.

3.2 Performance Analysis

Let $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ be a vector of the logical clock skews of all safe nodes at time t, where $x_i(t) = a_i\hat{a}_i(t)$. By multiplying both sides of (4) and (6) by a_i, we have $x_i(t^+) = \max\{x_i(t), x_j(t)\}$ and $x_i(t^+) = \max\{x_i(t), x_j^e(t)\}$ for $j \in \mathcal{V}_s$ and $j \in \mathcal{V}_a$, respectively, where $x_j^e(t) = a_j\delta_j^e(t)\hat{a}_j^e(t)$. Consider the following discrete time system for each safe node i as

$$x_i(k+1) = \begin{cases} \max\{x_i(k), x_j(k)\}, & j \in \mathcal{V}_s; \\ \max\{x_i(k), x_j^e(k)\}, & j \in \mathcal{V}_a, \end{cases} \tag{7}$$

where the state x_i represents the logical clock skew of node i. Then, the following theorem provides a sufficient and necessary condition for the convergence of the discrete time system (7), where the condition is also sufficient and necessary for

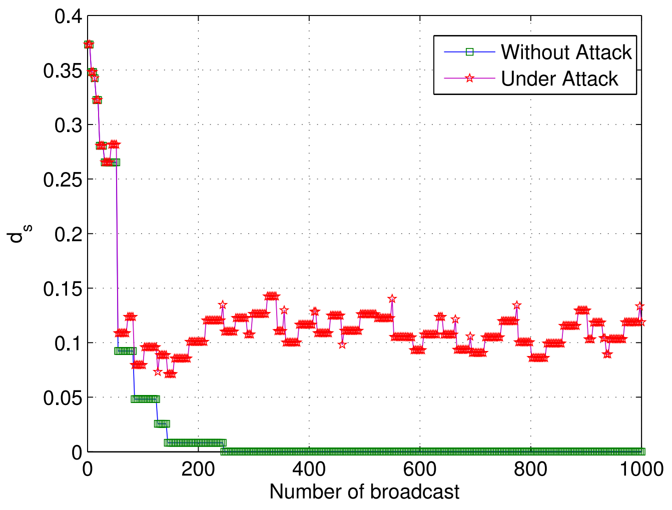


Fig. 1. Performance of MTS under attack.

that the skew compensation can be achieved as $x_i = a_i \hat{a}_i$ for each node i.

Theorem 3.1. *For the discrete time system (7),*

$$\lim_{k\to\infty} x(k) = c\mathbf{1}, \tag{8}$$

where c is a constant and $\mathbf{1} = [1, 1, \ldots, 1]^T$, holds iff. *there is a constant B such that*

$$\max_{j\in\mathcal{V}_a} \left\{ x_j^e(k) | k \in \mathbf{N}^+ \right\} \leq B. \tag{9}$$

Proof. The proof is given in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.150. □

It follows from Theorem 3.1 that if there exist attack nodes, which are able to make (9) violated, the skew compensation cannot be achieved. In fact, without any modification of protocols, each attack node j can break skew compensation easily. For example, since the hardware clock checking is not difficult [14], the attacker can set $\delta_j^e(t) = 1$, however, it can manipulate $\hat{a}_j^e(t)$ to destroy traditional MTS without being detected. For instance, if setting $\hat{a}_j^e(t) = \hat{a}_j^e(t-1) + \frac{1}{t}$, we have $\lim_{k\to\infty} x_j^e(k) = \infty$; if setting $\hat{a}_j^e(t)$ equal to the true value plus a positive random number, then $Prob\{\lim_{k\to\infty} x_j^e(k) = \infty\} = 1$. That is, both of them will make the maximal clock skew diverged.

In order to show the performance of MTS under message manipulations, we conduct simulation on a ring network with 30 nodes. Suppose at the first stage, all the nodes behave exactly according to the MTS protocol. However, at time 5, node 10 is compromised by the attacker and will broadcast $\hat{a}_{10} + \omega_{10}$ to its neighbor nodes in the following communications, where ω_{10} is randomly chosen from the interval $[0, 0.01]$. Let $d_s(t)$ be the maximum difference between the logical skews of any two safe nodes, i.e., $d_s(t) = \max_{i,j\in\mathcal{V}_s}\{x_i(t) - x_j(t)\}$. Fig. 1 shows the trajectories of $d_s(t)$. It can be observed that d_s will finally vary over an average value of around 0.1, which further indicates that the maximal logical clock difference would diverge in an approximately linear speed with a high probability. Apparently, a single node attack can deteriorate the performance of MTS in an easy way.

3.3 Design Challenges and Properties

Most of existing time synchronization protocols assume that all nodes are trustable. However, the existence of attack nodes requires to design an additional checking mechanism for preventing the manipulated information. There do exist some protocols focusing on the checking mechanism design but only for offset compensation. It should be pointed out that for pure offset compensation, each node only requires the neighbor hardware clock readings, which makes the information checking mechanism easy to be implemented, such as [14] and [15]. The key idea is to exploit the linearity of hardware clock readings to design checking mechanism. However, for the clock model which requires both skew and offset compensation, the problem becomes much more complicated as more parameters are required, e.g., $\hat{a}_i$ and $\hat{b}_i$. It increases the difficulty for safe node to detect the manipulated message, as the attack node i has more opportunities to attack the network as it can fake either the hardware clock reading or the parameters $\hat{a}_i$ and $\hat{b}_i$. Moreover, unlike the hardware clock reading, $\hat{a}_i$ and $\hat{b}_i$ depend on the implemented protocols, which increases the difficulty of checking mechanism design.

Despite of the challenges discussed above, it is observed that there are two important properties of MTS as follows, which can be exploited to design safeguard mechanisms. First, the hardware clock remains as a linear function of time t, which still can be utilized to design a hardware checking process as the safeguard mechanism of hardware clock. Second, note that in MTS each node will select the neighbor node with maximum logical clock as reference. Meanwhile, based on its own information, given a node i, it can calculate the values of $\hat{a}_j(t)$ and $\hat{b}_j(t)$ for its neighbor node j when node j selects node i as the reference node. Thus, $\hat{a}_j(t)$ and $\hat{b}_j(t)$ for node j can be calculated by one of its neighbor node i and included in the packet sent from node i. Since the information can be authenticated with MAC or digital signature, all nodes cannot modify the information received from neighbor nodes. This fact can be exploited to develop a logical clock checking process as the safeguard mechanism of logical clock. The details of the complete secure time synchronization protocol will be provided in the following section.

4 Secured Maximum Consensus Based Time Synchronization Protocol: SMTS

In this section, we will provide the details of secured maximum consensus based time synchronization protocol along with complete performance analysis. The overall architecture of SMTS is depicted in Fig. 2, which consists of six components. Since Message reception and verification, Message generation and authentication, Message broadcasting are common components for different protocols, we will focus on explaining the rest three components in detail as follows.

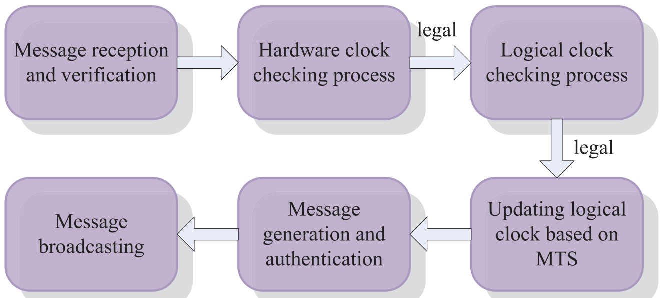

Message reception and verification
Hardware clock checking process
legal
Logical clock checking process
legal
Updating logical clock based on MTS
Message generation and authentication
Message broadcasting

Fig. 2. Overall architecture of SMTS.

4.1 Safeguard Mechanism of Hardware Clock

The hardware clock checking process, i.e., safeguard mechanism of hardware clock, is introduced as follows.

For $\forall i, j \in \mathcal{V}$, define $s_{ij}(k)$ as one-step relative skew estimation for node i with respect to node j,

$$s_{ij}(k) = \frac{\tau_j(t_k) - \tau_j(t_{k-1})}{\tau_i(t_k) - \tau_i(t_{k-1})}, \tag{10}$$

where k denotes kth of estimation, t_k is the corresponding real time. The following distributed algorithm RSE is used to estimate the relative skew of each neighbor pair of nodes.

Algorithm 1 : *Relative Skew Estimation (RSE)*

```
1: Set $\varepsilon_1 \geq 0$ to each node $i$, $i \in \mathcal{V}$.
2: Each node $i$ broadcasts its current hardware clock reading
   $\tau_i(t)$ to its neighbor nodes at each iteration.
3: For any node $i$'s neighbor node, say node $j$, upon receiving
   a message from node $i$, it records $(\tau_i(t), \tau_j(t))$.
4: If node $j$ successfully receives messages from node $i$ more
   than once, it then computes $s_{ij}(k)$ by (10).
5: For $\forall k > 1$, if $s_{ij}(k)$ satisfies
       $|s_{ij}(k) - s_{ij}(1)| \leq \varepsilon_1, \forall i \in \mathcal{N}_j,$     (11)
   then node $j$ assign $a_{ij}(k) = s_{ij}(k)$. Otherwise, it will deem
   node $i$ as the attack node, and break up with node $i$ by
   ignoring all its following messages.
```

Algorithm 1 utilizes the linear clock model to check the consecutive neighbor hardware readings at each time step, so that the attacker, if exists, cannot freely change the hardware clock reading for broadcasting.

Note that when measurement noise and communication delay are ignorable, and each hardware clock skew is a constant in (1), and thus $s_{ij}(k) = \frac{a_j}{a_i}$ holds for each integer k, we can set $\varepsilon_1 = 0$ to the nodes in Algorithm 1. For an attack node j, in order to avoid being identified by its neighbor safe nodes, it should ensure that (11) hold for $\varepsilon_1 = 0$. Thus, we have $s_{ij}(k+1) = s_{ij}(k)$, which yields

$$\frac{\tau_j^e(t_{k+1}) - \tau_j^e(t_k)}{\tau_i(t_{k+1}) - \tau_i(t_k)} = \frac{\tau_j^e(t_k) - \tau_j^e(t_{k-1})}{\tau_i(t_k) - \tau_i(t_{k-1})}. \tag{12}$$

Thus, the relative skew a_{ij}^e at each time step is guaranteed to be constant.

Note that

$$\begin{aligned} a_{ij}^e(t_k) &= \frac{\tau_j(t_k) - \tau_j(t_{k-1})}{\tau_i(t_k) - \tau_i(t_{k-1})} \frac{\tau_j^e(t_k) - \tau_j^e(t_{k-1})}{\tau_j(t_k) - \tau_j(t_{k-1})} \\ &= \frac{a_j}{a_i} \frac{\tau_j^e(t_k) - \tau_j^e(t_{k-1})}{\tau_j(t_k) - \tau_j(t_{k-1})}, \forall k \in \mathbf{N}^+, \end{aligned} \tag{13}$$

which implies that (12) holds *iff*

$$\frac{\tau_j^e(t_k) - \tau_j^e(t_{k-1})}{\tau_j(t_k) - \tau_j(t_{k-1})} = c_j, \forall k \in \mathbf{N}^+, \tag{14}$$

where c_j is a constant and satisfies $c_j = \frac{\tau_j^e(t_1) - \tau_j^e(t_0)}{\tau_j(t_1) - \tau_j(t_0)}$. Combining (14) with (13), it yields $a_{ij}^e = c_j \frac{a_j}{a_i}$. Since $\tau_j(t)$ is a linear function of real time t, it follows from (14) that $\tau_j^e(t_k) = c_j a_j t_k - c_j a_j t_0 + \tau_j^e(t_0)$ for $\forall, k \in \mathbf{N}^+$. Therefore, to avoid being identified by others, the hardware clock $\tau_j^e(t)$ for each attack node j broadcast at time t should satisfy

$$\tau_j^e(t) = a_j^e t + b_j^e, \qquad \forall k \in \mathbf{N}^+, \tag{15}$$

where $a_j^e = c_j a_j$ and $b_j^e = \tau_j^e(t_0) - c_j a_j t_0$.

Remark 4.1. The safe nodes will use the incorrect $\tau_j^e(t)$ for clock updating only when $\tau_j^e(t)$ satisfies (15), which is still a linear function of real time t. It is common to design a checking process to avoid the manipulation of hardware reading by exploring the relative skew estimation, e.g., [14]. However, each attack node can still use $\tau_j^e(t)$ to attack its neighbor nodes such that its neighbor node i gets incorrect relative skew $a_{ij}^e = \frac{a_j^e}{a_i} \neq \frac{a_j}{a_i}$, which leads to that node i updates the logical clock based on incorrect relative skew and selects $\tau_j^e(t)$ as the reference clock. Meanwhile, attack node j can decide c_j so that a_{ij}^e obtained by node i meets its requirement. For example, if attack node j selects $\tau_j^e(t) = c_j \tau_j(t)$ to broadcast, where $c_j < \min_{i \in \mathcal{N}_j} s_{ji}(k)$, then its neighbor node i will obtain a_{ij}^e which satisfies $a_{ij}^e = c_j \frac{a_j}{a_i} = \frac{c_j}{s_{ji}(k)} < 1$ for $\forall i \in \mathcal{N}_j$.

Remark 4.2. Taking noise, including measurement error, communication delay and the fluctuation of hardware clock skews, into consideration, each one-step relative skew estimation $s_{ij}(k)$ will fluctuate and not equal to a constant. Fortunately, the fluctuation of $s_{ij}(k)$ caused by the noise is usually small as the fluctuation of these noises are generally small, e.g., the variance of communication delay is about 10^{-8} [28]. Hence, we can set a small positive constant ε_1 in (11), such that the fluctuation of $s_{ij}(k)$ caused by these noises is bounded by ε_1 (see Example 4.3 as illustration). Under the constrains of (11), by similarly analysis as the above ideal case (noises are ignored), the hardware clock readings received from neighbor nodes is an approximately linear function, which can restrain the attack nodes from freely modifying hardware clock reading to attack. However, the synchronization accuracy is affected by the noises, the detailed analysis will be given in simulation section.

Example 4.3. Assume that the common broadcast period of all nodes is T and the hardware clock reading for each node i broadcast satisfies

$$\tau_i(t) = a_i t + b_i + a_i \theta_i(t), \tag{16}$$

where $a_i\theta_i(t)$ satisfied $|\theta_i(t)| \le \frac{\varepsilon}{2}$ is used to model the noise. For $\forall i, j \in \mathcal{V}_s$, substituting (16) into (10), it follows that $\frac{a_j}{a_i}\frac{T-\varepsilon}{T+\varepsilon} \le s_{ij}(k) \le \frac{a_j}{a_i}\frac{T+\varepsilon}{T-\varepsilon}$, and thus have $|s_{ij}(k) - s_{ij}(1)| < \frac{a_j}{a_i}\frac{4\varepsilon T}{T^2-\varepsilon^2}$. Hence, we can set $\varepsilon_1 = \frac{a_{max}}{a_{min}}\frac{4\varepsilon T}{T^2-\varepsilon^2}$ for (11), where $a_{max} = \max_{i\in\mathcal{V}} a_i$ and $a_{min} = \min_{i\in\mathcal{V}} a_i$.

4.2 Safeguard Mechanism of Logical Clock

This section describes the logical clock checking process, i.e., safeguard mechanism of logical clock. For simplifying the statements, in this section, $\hat{a}_i(t)$ and $\hat{b}_i(t)$ and $a_{ij}(t)$ are replaced by $\hat{a}_i$ and $\hat{b}_i$ and a_{ij} for $i, j \in \mathcal{V}$, respectively.

Note that if a node j selects node i's logical clock as the reference clock, the $\hat{a}_j$ and $\hat{b}_j$ used for the updates of node j should satisfy respectively

$$\hat{a}_j = \frac{\hat{a}_i}{a_{ij}} \tag{17}$$

and

$$\hat{b}_j = \hat{a}_i\tau_i(t_0) + \hat{b}_i - \hat{a}_j\tau_j(t_0), \tag{18}$$

where $\tau_i(t_0)$ and $\tau_j(t_0)$ are obtained in Algorithm 1. Thus, node i can calculate $\hat{a}_j$ and $\hat{b}_j$ respectively by using (17) and (18) based on the information held by itself. This fact is exploited to develop the logical clock checking process for SMTS. Before broadcasting, each node will authenticate the information so that all its neighbors can only read. Specifically, with the localized encryption and authentication protocol in [20], [23], each node will only share the reading key, which prevents neighbor nodes to manipulate the message.

Before presenting the details of logical clock checking process, we would like to first briefly define the communication format among sensor nodes. Define $\lambda^*_{ij} = [\lambda^*_{ij}(1), \lambda^*_{ij}(2)]$ as the authenticated message which is created by node i and used for broadcasting to its neighbor node j, where $\lambda^*_{ij}(1) = \hat{a}_j$ and $\lambda^*_{ij}(2) = \hat{b}_j$ are obtained from (17) and (18). In order to run the logical clock checking process, let the packet for node i broadcasting should include λ^*_{ij} and λ^*_{li}, where λ^*_{li} is the message received from a neighbor node l by node i, and $\lambda^*_{li}(1)$ and $\lambda^*_{li}(2)$ are respectively equal to the current adjusting parameters used for the node i's logical clock. If node i has not yet updated its logical clock based on λ^*_{li} for $\forall l \in \mathcal{N}_i$ and $l \ne i$, let $\lambda^*_{li} = \lambda^*_{ii} = [1, 0]$ and use λ^*_{ii} for broadcasting, i.e., $\lambda^*_{li} = [1, 0]$ for $l = i$.

Now, the key step of logical clock checking process is provided, which prevents the attack node i from freely using incorrect $\hat{a}_i$ and $\hat{b}_i$ to attack. That is, when node j receives the information from node i and selects node i's logical clock as the reference clock, it checks whether the following two equations hold true or not,

$$|\lambda^*_{ij}(1) - \lambda^*_{li}(1)a_{ji}| \le \varepsilon_2, \tag{19}$$

where $\varepsilon_2 \ge 0$, and

$$|\lambda^*_{ij}(2) + \lambda^*_{ij}(1)\tau_j(t_1) - \lambda^*_{li}(1)\tau_i(t_1) - \lambda^*_{li}(2)| \le \varepsilon_3, \tag{20}$$

where $\tau_i(t_1)$ and $\tau_j(t_1)$ are also obtained in Algorithm 1 and $\varepsilon_3 \ge 0$. Note that $\lambda^*_{li}(1) = \hat{a}_i$, $\lambda^*_{lj}(2) = \hat{a}_j$, $\lambda^*_{ij}(1) = \hat{a}_j$ and $\lambda^*_{ij}(2) = \hat{b}_j$ are obtained from (17) and (18). Each a_{ij} is estimated by (10). Substituting these equations into the left sides of both (19) and (20) yields two functions of $\tau_i(t)$, $\tau_j(t)$ and $\hat{a}_i$. We thus can calculate the lower and upper bound of these two functions when the noise model and bound are given. Then, we can select suitable ε_2 and ε_3 for (19) and (20), respectively.

Since the noises are omitted, we set $\varepsilon_2 = \varepsilon_3 = 0$ for (19) and (20). When the above two equations are both true, node j will trust the node i; otherwise, node i will be thought as an attacker by the node j. Note both (19) and (20) hold *iff.* the parameters $\hat{a}_i$ and $\hat{b}_i$ used in (17) and (18) satisfy $\hat{a}_i = \lambda^*_{li}(1)$ and $\hat{b}_i = \lambda^*_{li}(2)$ (where λ^*_{li} should satisfy $\lambda^*_{ii} = [1, 0]$ for $l = i$). Meanwhile, the right sides of both (19) and (20) cannot be modified by the node i as λ^*_{li} is authenticated by neighbor node l and $\tau_i(t_1)$ should have passed the hardware clock checking process. Thus, (19) and (20) guarantee that the λ^*_{ij} created by an attack node i for transmitting to node j cannot be freely decided by node i itself.

From the above, the logical clock checking process guarantees that node j updates it logical clock based on correct λ^*_{ij}, which is received and created by the neighbor node i. Therefore, logical clock checking process designed for SMTS ensures that all safe nodes will not use incorrect adjusting parameters for clock updates.

4.3 SMTS Protocol

In SMTS, after the received messages pass the hardware clock and logical clock checking processes, the nodes will update their logical clock based on MTS. The details of SMTS are introduced as follows.

For energy saving, node i will broadcast only when it finds that there is at least one q_{ij} satisfying $q_{ij} > 1$ for $j \in \mathcal{N}_i$. Assume that nodes j and l are in $\mathcal{N}_i$, where node l is the current reference node of node i. The detailed SMTS is depicted in Algorithm 2.

Algorithm 2 : *The SMTS Protocol*

```
1: Set initial conditions as $\hat{a}_i = 1$, $\hat{b}_i = 0$ and $\lambda^*_{ij} = [\lambda^*_{ij}(1), \lambda^*_{ij}(2)] = [1, 0]$ for each node $i$ and for each its neighbor node $j$.
2: Apply RSE to estimate the relative skews for each node and its neighbor nodes.
3: At each iteration, node $i$ computes $q_{ij}$ by $q_{ij}(t) = \frac{a_{ij}(t)\hat{a}_j(t)}{\hat{a}_i(t)}$.
4: If $q_{ij} > 1$, node $i$ calculates $\lambda^*_{ij}(1) = \frac{\hat{a}_i}{a_{ij}}$ and $\lambda^*_{ij}(2) = \hat{a}_i\tau_i(t_0) + \hat{b}_i - \lambda^*_{ij}(1)\tau_j(t_0)$, and then creates $\lambda^*_{ij}$, where $\tau_i(t_0)$ and $\tau_j(t_0)$ are stored in node $i$, which were obtained from RSE.
5: Broadcast $\lambda^*_{li}$ and $\lambda^*_{ij}$ to node $j$, where $\lambda^*_{li}$ was received from node $l$ and should satisfy $\lambda^*_{li} = [\hat{a}_i, \hat{b}_i]$ (where $\lambda^*_{li} = [1, 0]$ for $l = i$).
6: If $\lambda^*_{ij}(1) > \hat{a}_j$, node $j$ computes $\lambda^*_{li}(1)a_{ji}$ and $\lambda^*_{li}(1)\tau_i(t) + \lambda^*_{li}(2) - \lambda^*_{ij}(1)\tau_j(t)$.
7: If both (19) and (20) are true,
       $\hat{a}_j = \lambda^*_{ij}(1)$, $\hat{b}_j = \lambda^*_{ij}(2)$.
   Then, node $j$ stores $\lambda^*_{ij}$ and $\hat{a}_i = \lambda^*_{li}(1)$.
8: If both (19) and (20) or one of them is false, then node $j$ will regard node $i$ as an attack node and will no longer use the information received from node $i$ for logical clock updates.
```

Remark 4.4. In SMTS, $\hat{a}_i$ and $\hat{b}_i$ of node i for broadcasting cannot be modified by itself due to that they are obtained from a neighbor node l and included in the authenticated message. Meanwhile, based on equations (19) and (20), the neighbor node j can detect whether the node i transmits correct $\hat{a}_j$ and $\hat{b}_j$ to it or not, which helps the safe node to avoid using incorrect parameters to adjust logical clock. The checking process will only be valid when node i can estimate the $\hat{a}_j$ and $\hat{b}_j$ of each neighbor node j based on its current $\hat{a}_i$ and $\hat{b}_i$ without the knowledge of the adjusting parameters of node j, which is indeed a key characteristic of maximum consensus concept.

Remark 4.5. For SMTS, due to the hardware clock and logical clock checking process, each attack node j can successfully attack its safe neighbors only by one incorrect $\tau_j^e(t)$ with constant a_j^e and b_j^e. Once the attack node j has used more than one different $\tau_j^e(t)$ or incorrect adjusting parameters to attack the safe nodes, it will be detected and isolated by the safe neighbor nodes, which means that all these attacks are invalid. Thus, we say all attack nodes have finished their attacks when every attack node j has attacked its neighbor nodes by one linear $\tau_j^e(t)$.

4.4 Convergence Analysis

Note that SMTS protocol has guaranteed that the hardware clock $\tau_j^e(t)$ used by a neighbor node i is a linear function of real time t, and the attack node j cannot manipulate the logical clock parameters $\hat{a}_j$ and $\hat{b}_j$. Therefore, only the incorrect hardware clock reading $\tau_j^e(t_k)$ can be used by each attack node j to attack the algorithm, and the $\tau_j^e(t)$ should satisfy (15). Thus, we just need to analyze the convergence problem for this situation. Note that the hardware clocks of the whole network can be described as $\tau_i(t) = a_i t + b_i$ for $i \in \mathcal{V}_s$ and $\tau_j^e(t) = a_j^e t + b_j^e$ for $j \in \mathcal{V}_a$.

To achieve the purpose of attack, each attacker will select $\tau_j^e(t)$ such that a_j^e is larger than all logical clock skews of its neighbor nodes. Since attackers cannot collude, $a_j^e, j \in \mathcal{V}_a$, selected by attackers are usually different from each other and they are also usually different from $a_i, i \in \mathcal{V}_s$. Assume that a_j^e $(j \in \mathcal{V}_a)$ and a_i $(i \in \mathcal{V}_s)$ are different from each other. Since each attack node j can only use one linear $\tau_j^e(t)$ to attack to avoid being detected, there are at most $m+n$ different clocks in the whole network.

Define $a_{max} = \max\{\max_{i\in\mathcal{V}_s} a_i, \max_{j\in\mathcal{V}_a} a_j^e\}$. Assume that node $c, c \in \mathcal{V}$, is the node whose hardware clock skew is equal to a_{max} at time t_0, i.e., $a_c = a_{max}$ (or $a_c^e = a_{max}$), and its hardware clock $\tau_v(t)$ satisfies $\tau_c(t) = a_c t + b_c$. Let $\mathcal{V}_c(t)$ be a subset of $\mathcal{V}_s$, i.e., $\mathcal{V}_c(t) \subseteq \mathcal{V}_s$, and the logical clock skew and offset of each node in $\mathcal{V}_c(t)$ are equal to a_c and b_c at time t, respectively. The function $f(t)$ denotes the number of node belonging to the set $\mathcal{V}_c(t)$ at time t, i.e., $f(t) = |\mathcal{V}_c(t)| \geq 0$, where $|\mathcal{V}_c(t)|$ denotes the number of elementals in $\mathcal{V}_c(t)$. Since the initial condition satisfies $\hat{a}_i(0) = 1$ and $\hat{b}_i(0) = 1$ for $i \in \mathcal{V}$ in algorithm SMTS, it follows from the definition of $\mathcal{V}_c(t)$ that $f(0) = 1$ for $c \in \mathcal{V}_s$ and $f(0) = 0$ for $c \notin \mathcal{V}_s$. In the remaining parts of this paper, we say two nodes have the same clock, which means that the clock skew and clock offset of their logical clock are identical.

Lemma 4.6. $f(t) = n$ iff. $L_i(t) = \tau_c(t)$ *for* $\forall i \in \mathcal{V}_s$.

Proof. The proof is given in the supplementary file, available online. □

Theorem 4.7. *Suppose that the network* G_s *is connected and all attack nodes have finished their attacks before iteration* k_0 $(k_0 \in \mathcal{N}^+)$. *By using SMTS, the skew and offset of safe node* $i, i \in \mathcal{V}_s$, *converge to*

$$\begin{cases} \hat{a}_i(k_0+k)a_i = a_c, \\ \hat{a}_i(k_0+k)b_i + \hat{b}_i(k_0+k) = b_c, \end{cases} \tag{21}$$

for $\forall k \geq n-1$.

Proof. The proof is given in the supplementary file, available online. □

Remark 4.8. According to Theorem 4.7, once there is a safe node updating its logical clock such that it is equivalent to the hardware clock of node c at iteration k', at iteration $k'+n-1$ all safe nodes have updated their logical clocks such that they are the same as the hardware clock of node c. Hence, if the node c is a safe node, we obtain that $L_i(k) = \tau_c(t), i \in \mathcal{V}_s$, holds for $k \geq n-1$, which means that the convergence speed of SMTS is irrelevant to k_0 and the attacks of the attack nodes are ineffective. Meanwhile, only when the clock skew a_j^e of τ_j^e for attack node j attacking at time t is larger than all logical clock skew of safe nodes, i.e., $a_j^e > \max_{i\in\mathcal{V}_s} \hat{a}_i(t)a_i$, the attack of node j may affect the convergence speed of SMTS.

4.5 Communication Energy Cost

Communication energy cost is a major concern for WSN, which can be roughly estimated by the broadcasting times throughout the network. In Theorem 4.7, we have obtained the convergence speed of the algorithm SMTS. Assume that every broadcast of all the nodes costs the same amount of energy E and let E_c be the total energy cost for the synchronization algorithm to convergence. We omit the detailed analysis of energy cost for authentication process due to the space limitation as it has been presented in existing literature, e.g., [20].

Note from Theorem 4.7 that SMTS will converge in $n-1$ iterations if there is no attack or after one attack, and each attack node at most can attack the network and destroy the clock synchronization once. After the convergence has been reached, each safe node will know that all its neighbor nodes have the same logical clock after one broadcasts again, and then it will no longer broadcast information until it is attacked by the attack nodes. Hence, there are at most $(m+1)n$ iterations that all the safe nodes need to broadcast, which means that the total energy cost of these nodes is at most $E(m+1)n^2$. Additionally, for each safe node i, the initial three broadcasts are used for RSE, which is to estimate the relative skews, thus each safe node should cost $3E$ for RSE, i.e., all safe nodes need $3nE$ energy cost for RSE before the iteration starting for SMTS. Hence, we have $E_c \leq E[(m+1)n^2 + 3n]$. Then, we give an upper bound for E_c to SMTS as follows:

$$E_c \leq E[(m+1)n^2 + 3n], \tag{22}$$

where m denotes the number of attack nodes.

Remark 4.9. If there are no attacks in the network, we have $E_c \leq E[n^2 + 3n]$, which is the same as that of MTS. Note that the upper bound in (22) is an increasing function of attack node number m, which means that more attacks will lead to more energy cost for re-synchronization.

5 Simulation

Throughout the simulation examples, we set $\hat{a}(0) = 1$, $\hat{b}(0) = 0$ and $T = 1$, and let each skew a_i of the hardware clock be randomly selected from the interval $[0.8, 1.2]$ and the offset b_i of node i be randomly selected from the interval $[0, 0.4]$. For each iteration k, let d_{max} and D_{max} be variables, which are measured by the maximum difference of the logical clocks for safe nodes and all nodes, respectively, and satisfy $d_{max} = \frac{\max_{i,j\in\mathcal{V}_s}\{L_i(k)-L_j(k)\}}{k}$ and $D_{max} = \frac{\max_{i,j\in\mathcal{V}}\{L_i(k)-L_j(k)\}}{k}$. It is clear that all safe nodes have the same logical clock iff $d_{max} = 0$. All the following simulations are conducted in Matlab.7.0.

5.1 When Noises are Ignorable

Consider the ring network with 30 nodes, where node 10 is an attack node, which is the same as the case considered in Section 3. Assume that node 10 broadcasts the logical skew adjusting parameter with

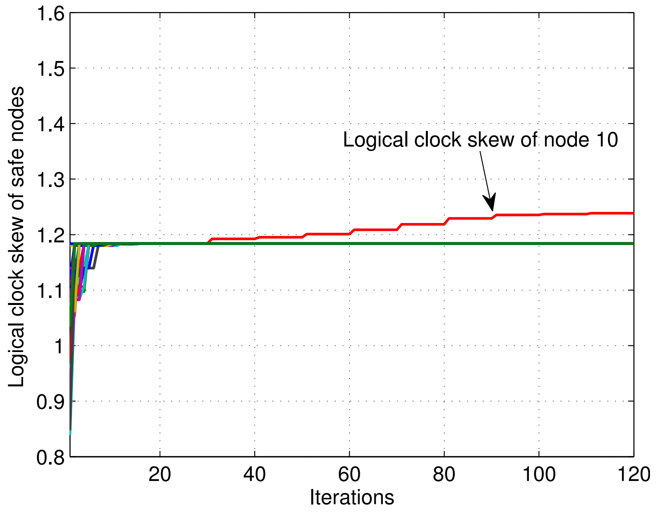
$$\hat{a}_{10}(t) = \hat{a}_{10}(t) + \omega_{10}(t), \tag{23}$$

where the ω_{10} is randomly selected in $[0, 0.01]$. Fig. 3a shows that the logical clock skews of all nodes change over iterations via SMTS, where the red line is the logical clock skew of node 10. Clearly, all safe node will converge and only the logical clock skew of node 10 becomes larger at each attack time of itself, which means that node 10 has been detected by its safe neighbor nodes. Then, the associate maximum logical clock differences d_{max} and D_{max} for safe nodes and all nodes are shown in Fig. 3b, respectively, which again shows that the logical clocks of safe nodes will converge. Hence, SMTS algorithm can effectively avoid the message manipulation attacks initiated by the attack node.

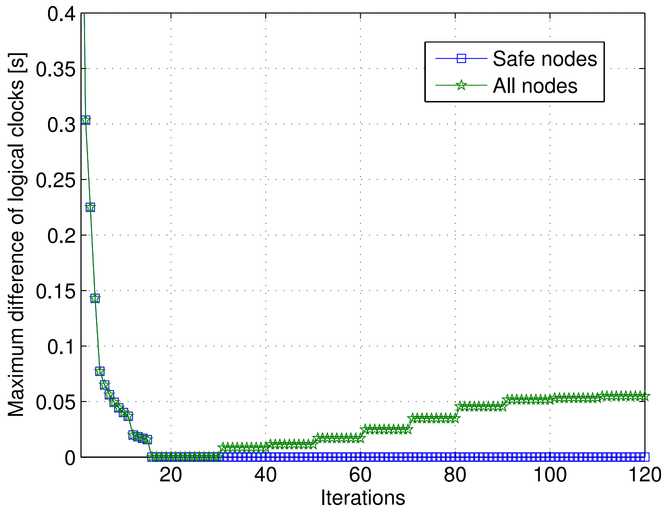
Generally, consider a random graph with $n = 100$ and $m = 5$, which means that there are 100 safe nodes and five attack nodes in the network. Let these nodes be randomly deployed in an 100×100 square meter area and the maximum communication range of each node is 20 meter. Let j_l be an attack node for $l = 1, 2, \ldots, m$, and assume that the attack time of each attack node j_l is equal to $25 \times l$ and the associate attack $\tau^e_{j_l}(t)$ satisfies

$$\tau^e_{j_l}(t) = c_j \tau_{j_l}(t), \tag{24}$$

where $c_j = \max_{i\in\mathcal{N}_j} \hat{a}_i s_{ji}(k) + 0.05\theta$ and θ is a random number selected in $[0, 1]$. The profiles of the logical clock skews of safe nodes over iterations of SMTS are shown in Fig. 4a. It is observed that all safe nodes' logical clock skews converge under SMTS in less than 100 iterations initially, and re-converge in less than 100 iterations after attack. Then, the profiles of the associate maximum difference of the logical clocks for each node is shown in Fig. 4b. Comparing Fig. 4a with Fig. 4b, it is observed that both of them converge at the same time, which implies that the compensation of clock skew and offset are finished at the same time. For comparison, in Fig. 5, we also show the performance of SMTS under five malicious nodes which launch their attacks at the same time, e.g., iteration 50. It can be observed that divided attacks, e.g., Fig. 5, degrade the SMTS more seriously than the attacks launching at the same time, which also supports the results of Theorem 4.7.



(a) Logical clock skews of all nodes.



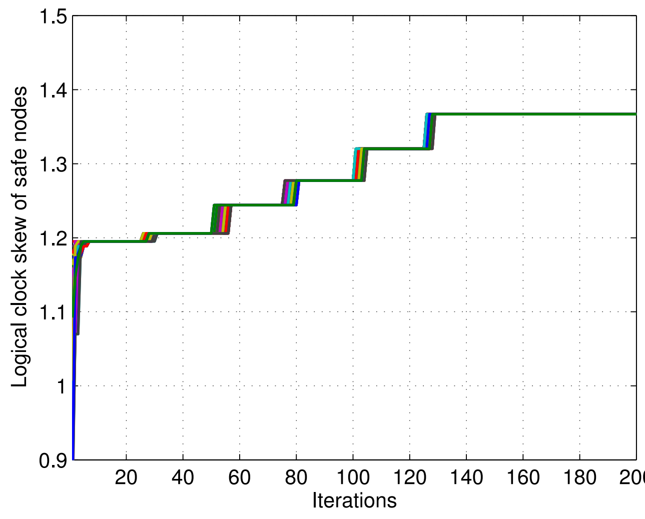
(b) Maximum differences of d_{max} and D_{max}.

Fig. 3. The performance of SMTS under attacks (23).

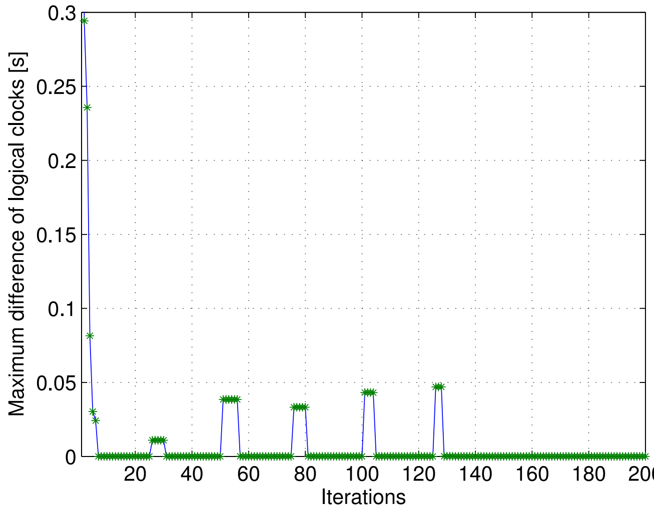
Then, for the random graph defined above which has 100 safe nodes, let the number of attack nodes m change from 0 to 20, and assume the attack time of each attack node j_l $(l = 1, 2, \ldots, m)$ is equal to $25 \times l$ and the associate $\tau^e_{j_l}(t)$ satisfies (24). The relation between the convergence time of SMTS and the number of attack nodes is given in Fig. 6. It is observed that if attackers launch their attacks as described in (24), the convergence time of SMTS is approximately linearly correlated with the number of attack nodes, m.

5.2 When Noises Are Considered

In this section, we study the performance of SMTS under attack with different noises in the random graph defined in



(a) Logical clock skews of safe nodes.



(b) Maximum differences of d_{max}.

Fig. 4. The performance of SMTS under attacks (24).

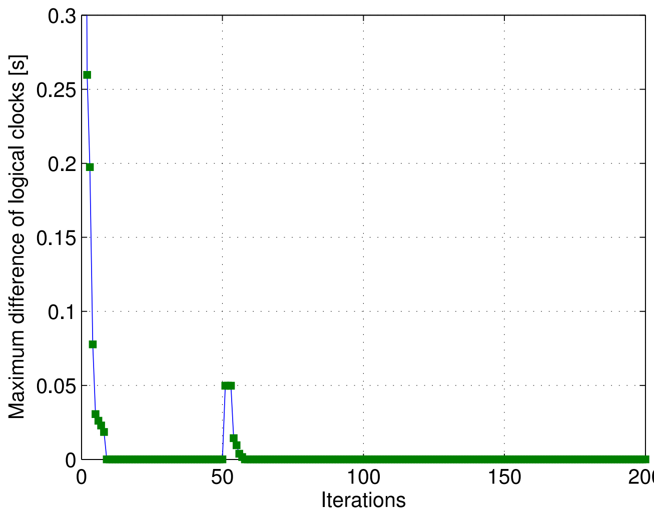


Fig. 5. The performance of SMTS when five attack nodes launch attacks at the same time.

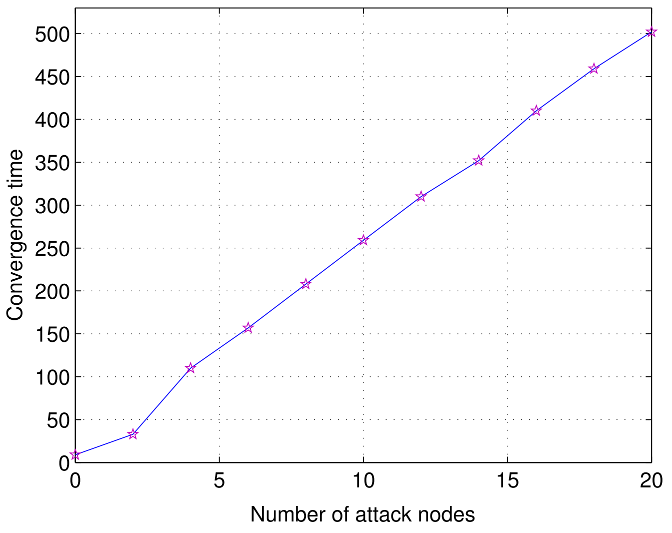


Fig. 6. The relation between the convergence time of SMTS and the number of attack nodes.

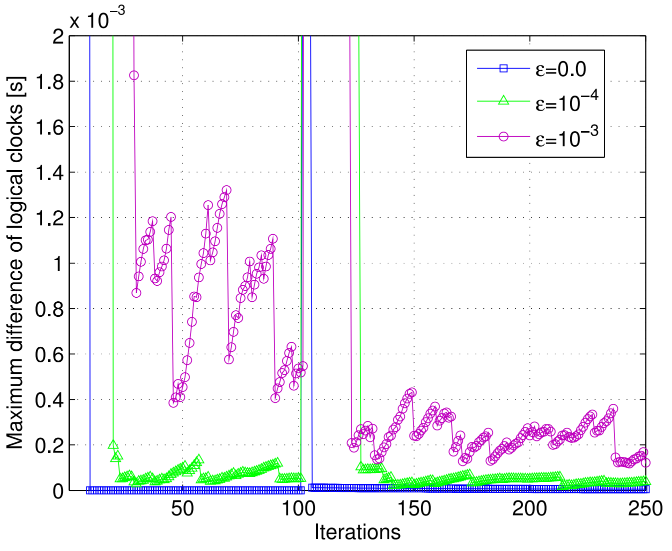


Fig. 7. The performance of SMTS under attacks (25) with different noises bound setting.

the above, including 100 safe nodes and 5 attack nodes. Consider the same noise model as (16) given in Example 4.3 and the following attack strategies:

$$\tau^e_{j_l}(t) = c_j(t)\tau_{j_l}(t) + a_{j_l}\theta_{j_l}(t), \qquad (25)$$

where c_j is the same as in (24) and $\theta_{j_l}(t)$ is a random number selected from $[-12\varepsilon, 12\varepsilon]$. Clearly, attack strategy (25) is more general than (24). In the following simulations, set $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 6\varepsilon$ for SMTS, which can guarantee that each safe node will not be isolated by other safe nodes. Let each attacker begins its attack at $t = 100$ with (25). The information of attackers are creditable only when $\theta_{j_l}(t)$ is bounded by 6ε. The performance of SMTS under different ε setting is shown as in Fig. 7. Clearly, SMTS can still converge under different noise bound setting (where the values between iterations 100 and 120 are less than 0.02) while the synchronization accuracy deceases with the noise bound, and the perfect synchronization can be achieved only when there is no noise, i.e., when $\varepsilon = 0$.

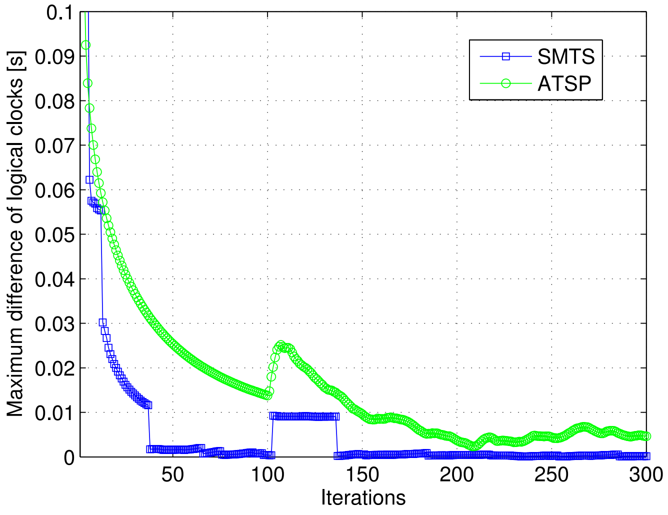


Fig. 8. The performance of SMTS and ATSP under attacks (25).

Fig. 8 compares the performance of SMTS and ATSP (a distributed secure protocol proposed in [14]) under attacks (25) with setting $\varepsilon = 10^{-3}$ for SMTS and the minimum error threshold $e_{min} = 10^{-3}$ for ATSP, and the other parameters of ATSP are the same as the setting in [14]. It is observed that SMTS provides much faster convergence speed and better synchronization accuracy in this scenario.

6 Related Works

In the literature, different efforts have been devoted to providing secure time synchronization services for WSN [13], where most of them focus on enhancing existing protocols by using authentication and fault detection mechanisms.

Du et al. achieve time synchronization by introducing high power nodes to form hierarchical topologies [17]. However, they use very simple clock model without considering skew errors. By checking if the end-to-end delays exceed some prescribed threshold, Ganeriwal et al. propose a protocol suite to secure both pairwise and group-wise synchronization [19]. Sun et al. establish a level hierarchy and allow synchronized node diffuse its clock to the network to achieve network-wide synchronization [18]. Chiang et al. in [24] present secure time synchronization protocol for WSNs under a man-in-the-middle attack, where the attacker could prevent the proper operation of the clock synchronization protocol. Rahman et al. propose a protocol in [16], which uses pairing and identity-based cryptography to secure the time synchronization to reduce the communication and storage requirements of each node. Huang et al. propose several techniques to reinforce the structure of FTSP to defend against attacks from malicious nodes [15]. However, most of these protocols rely on some reference clocks, which are vulnerable to intelligent attacks.

Hu et al. propose a distributed and secure synchronization protocol ATSP that can tolerate attacks of node compromising, packet faking and delaying [14]. They point out that these three attacks are equivalent to falsifying the time-stamps of the clock packets. ATSP is able to accurately detect attacks and iteratively achieve synchronization across the network in a fully distributed manner. Nevertheless, the clock skew errors are not compensated. Moreover, ATSP only promises bounded ultimate synchronization error.

7 Conclusion

This paper investigates time synchronization under cyber physical attacks in WSNs. The theoretical analysis and simulation results are given to show that the existing maximum consensus based time synchronization protocol is invalid under message manipulation attacks defined in this paper. A novel secured maximum consensus based time synchronization protocol is proposed to defend against message manipulation attacks. Specifically, in SMTS, by carefully designing the hardware clock and logical clock checking processes, it will be able to detect and invalidate the potential message manipulation attacks. Meanwhile, the maximum consensus based logical clock updating process guarantees the fast convergence and compensates clock skew and offset simultaneously. Extensive simulations demonstrate the effectiveness of SMTS. Future directions include handle more attack strategies of attack sensor nodes and experimental validation of the results.

Acknowledgments

This work was supported in part by the NSFC under Grants 61004060, 61222305, the 863 High-Tech Project under Grant 2011AA040101-1, the SRFDP under Grants 20100101110066, 20120101110139, NCET-11-0445, and the Fundamental Research Funds for the Central Universities under Grants 2013QNA5013 and 2013FZA5007. Jiming Chen is the correspondence author.

References

[1] B. Sundararaman, U. Buy, and A.D. Kshemkalyani, “Clock Synchronization for Wireless Sensor Networks: A Survey,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 281-323, 2005.

[2] Q. Li and D. Rus, “Global Clock Synchronization in Sensor Networks,” *Proc. IEEE INFOCOM*, pp. 564-574, 2004.

[3] A. Abdulla, H. Nishiyama, J. Yang, N. Ansari, and N. Kato, “HYMN: A Novel Hybrid Multi-Hop Routing Algorithm to Improve the Longevity of WSNs,” *IEEE Trans. Wireless Comm.*, vol. 11, no. 7, pp. 2531-2541, July 2012.

[4] J. Chen, Q. Yu, Y. Zhang, H. Chen, and Y. Sun, “Feedback-Based Clock Synchronization in Wireless Sensor Networks: A Control Theoretic Approach,” *IEEE Trans. Vehicular Technology*, vol. 59, no. 6, pp. 2963-2973, July 2010.

[5] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, “The Flooding Time Synchronization Protocol,” *Proc. Second Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.

[6] S. Philipp and W. Roger, “Gradient Clock Synchronization in Wireless Sensor Networks,” *Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2009.

[7] L. Schenato and F. Fiorentin, “Average Timesynch: A Consensus-Based Protocol for Time Synchronization in Wireless Sensor Networks,” *Automatica*, vol. 47, no. 9, pp. 1878-1886, 2011.

[8] J. He, P. Cheng, L. Shi, and J. Chen, “Time Synchronization in WSNs: A Maximum Value Based Consensus Approach,” *Proc. 50th IEEE Conf. Decision and Control and European Control Conf. (CDC-ECC)*, pp. 7882-7887, 2011.

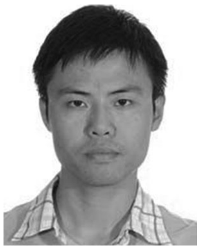
[9] B. Choi, H. Liang, X. Shen, and W. Zhuang, “DCS: Distributed Asynchronous Clock Synchronization in Delay Tolerant Networks,” *IEEE Trans. Parallel and Distributed Systems*, vol. 23, no. 3, pp. 491-504, Mar. 2012.

[10] A. Cardenas, S. Amin, and S. Sastry, “Research Challenges for the Security of Control Systems,” *Proc. Third Conf. Hot Topics in Security (HotSec '08)*, 2008.

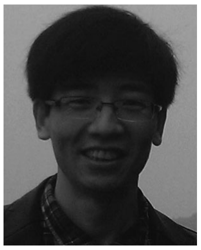
[11] D. Boyle and T. Newe, "Securing Wireless Sensor Networks: Security Architectures," *J. Networks*, vol. 3, no. 1, pp. 65-76, 2008.
[12] M. Valero, S. Jung, A. Uluagac, Y. Li, and G. Atlanta, "Di-Sec: A Distributed Security Framework for Heterogeneous Wireless Sensor Networks ," *Proc. IEEE INFOCOM*, 2012.
[13] K, Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks," *Proc. ACM Conf. Computer and Comm. Security (CCS)*, 2006.
[14] X. Hu, T. Park, and K.G. Shin, "Attack-Tolerant Time-Synchronization in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2008.
[15] D. Huang, K. You, and W. Teng, "Secured Flooding Time Synchronization Protocol," *Proc. IEEE Eighth Int'l Conf. Mobile Adhoc and Sensor Systems (MASS)*, pp. 620-625, 2011.
[16] M. Rahman and K. El-Khatib, "Secure Time Synchronization for Wireless Sensor Networks Based on Bilinear Pairing Functions," *IEEE Trans. Parallel and Distributed Systems*, DOI: 10.1109/TPDS.2010.94, 2010.
[17] X. Du, M. Guizani, Y. Xiao, and H-H. Chen, "Secure and Efficient Time Synchronization in Heterogeneous Sensor Networks," *IEEE Trans. Vehicular Technology*, vol. 57, no. 4, pp. 2387-2394, July 2008.
[18] K. Sun, P. Ning, and C. Wang, "Secure and Resilient Clock Synchronization in Wireless Sensor Networks," *IEEE J. Selected Areas in Comm.*, vol. 24, no. 2, pp. 395-408, Feb. 2006.
[19] S. Ganeriwal, C. Popper, S. Capkun, and M.B. Srivastava, "Secure Time Synchronization in Sensor Networks," *ACM Trans. Information and Systems Security*, vol. 11, no. 4, article 23, 2008.
[20] R. Wang, W. Du, X. Liu, and P. Ning, "ShortPK: A Short-Term Public Key Scheme for Broadcast Authentication in Sensor Networks," *ACM Trans. Sensor Networks*, vol. 6, article 9, 2009.
[21] J. Sen, "A Survey on Wireless Sensor Network Security," *Int'l J. Comm. Networks and Information Security*, vol. 1, no. 2, pp. 59-82, 2009.
[22] Z. Yu and Y. Guan, "A Dynamic En-Route Filtering Scheme for Data Reporting in Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 18, no. 1, pp. 150-163, Feb. 2010.
[23] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *ACM Trans. Sensor Networks*, vol. 2, no. 4, pp. 500-528, 2006.
[24] J. Chiang, J. Haas, Y-C. Hu, P.R. Kumar, and J. Choi, "Fundamental Limits on Secure Clock Synchronization and Man-in-the-Middle Detection in Fixed Wireless Networks," *Proc. IEEE INFOCOM*, pp. 1962-1970, 2009.
[25] R. Lu, X. Lin, H. Zhu, X. Liang, and X. Shen, "BECAN: A Bandwidth-Efficient Cooperative Authentication Scheme for Filtering Injected False Data in Wireless Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 23, no. 1, pp. 32-43, Jan. 2012.
[26] A. Giridhar and P.R. Kumar, "Distributed Clock Synchronization over Wireless Networks: Algorithms and Analysis," *Proc. 45th IEEE Conf. Decision and Control (CDC)*, 2006.
[27] Y. Zhang, R. Yu, W. Yao, S. Xie, Y. Xiao, and M. Guizani, "Home M2M Networks: Architectures, Standards, and QoS Improvement," *IEEE Comm. Magazine*, vol. 49, no. 4, pp. 44-52, Apr. 2011.
[28] Z. Zhong, P.P. Chen, and T. He, "On-Demand Time Synchronization with Predictable Accuracy," *Proc. IEEE INFOCOM*, 2009.

Jianping He is currently working toward the PhD degree in control science and engineering at Zhejiang University, Hangzhou, China. He is a member of the Group of Networked Sensing and Control (IIPC-nesC) in the State Key Laboratory of Industrial Control Technology at Zhejiang University. His research interests include time synchronization, consensus, and distributed security algorithm design problems in wireless sensor networks.

Jiming Chen (M'08-SM'11) received the BSc and PhD degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2000 and 2005, respectively. He was a visiting researcher at INRIA in 2006, National University of Singapore in 2007, and University of Waterloo from 2008 to 2010. He is currently a full professor with the Department of Control Science and Engineering, and the coordinator of group of Networked Sensing and Control in the State Key laboratory of Industrial Control Technology, Vice Director of Institute of Industrial Process Control at Zhejiang University. He currently serves associate editors for several international journals including *IEEE Transactions on Parallel and Distributed System*, *IEEE Transactions on Industrial Electronics*, *IEEE Network*, IET Communications, etc. He was a guest editor of the *IEEE Transactions on Automatic Control*, *Computer Communication* (Elsevier), *Wireless Communication and Mobile Computer* (Wiley) and *Journal of Network and Computer Applications* (Elsevier). He also served/serves as Ad hoc and Sensor Network Symposium Co-Chair, IEEE GLOBECOM 2011; general symposia Co-Chair of ACM IWCMC 2009 and ACM IWCMC 2010, WiCON 2010 MAC track Co-Chair, IEEE MASS 2011 Publicity Co-Chair, IEEE DCOSS 2011 Publicity Co-Chair, IEEE ICDCS 2012 Publicity Co-Chair, IEEE ICCC 2012 Communications QoS and Reliability Symposium Co-Chair, IEEE SmartGridComm, The Whole Picture Symposium Co-Chair, IEEE MASS 2013 Local Chair, Wireless Networking and Applications Symposium Co-Chair, IEEE ICCC 2013 and TPC member for IEEE ICDCS '10, '12, '13, IEEE MASS '10,A '11, '13, IEEE SECON '11, '12, IEEE INFOCOM '11, '12, '13, etc.



Peng Cheng (M'10) received the BE degree in automation, and the PhD degree in control science and engineering in 2004 and 2009 respectively, both from Zhejiang University, Hangzhou, P.R. China. He is currently an associate professor with the Department of Control Science and Engineering, Zhejiang University. His research interests include networked sensing and control, cyber-physical system, and robust control.



Xianghui Cao (S'08-M'11) received the BS and PhD degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. During 2008-2010, he was a visiting scholar in the Department of Computer Science, The University of Alabama. He is currently a postdoctoral fellow in the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago. His research interests include networked estimation and control, wireless network performance analysis, and network security.