SATS: Secure Average-Consensus-Based Time Synchronization in Wireless Sensor Networks

Jianping He, Peng Cheng, *Member, IEEE*, Ling Shi, and Jiming Chen, *Senior Member, IEEE*

***Abstract*—It is important and challenging to achieve secure time synchronization in wireless sensor networks, which may be deployed in a hostile environment under various malicious attacks. The recently developed average-consensus-based time synchronization protocol (ATS) is a promising alternative as it does not depend on any reference node or network topology, which makes it robust to different kinds of attacks, e.g., denial-of-service, node destruction, etc. However, the in-network information fusion nature of average consensus makes the ATS vulnerable to the well-known message manipulation attacks. In this paper, we focus on how to defend the ATS protocol in wireless sensor networks under message manipulation attacks. We first investigate the impact of message manipulation attacks over ATS, and derive a necessary condition for ATS to converge. Then based on the obtained insights, we propose a novel adjusting parameter checking mechanism which exploits the two-hop neighboring information to dynamically constrain the attackers. We further incorporate all checking processes into the traditional ATS protocol to form a secure average-consensus-based time synchronization protocol (SATS). We prove that SATS guarantees the network time synchronization with an exponentially converging speed.**

***Index Terms*—Wireless sensor networks, time synchronization, average consensus; security, message manipulation attack.**

I. Introduction

TIME synchronization is a crucial requirement for various applications in wireless sensor networks (WSNs), e.g., data aggregation, target tracking, scheduling and sensor nodes cooperation [1]. Different protocols have been proposed for time synchronization in various scenarios [2]–[4], [6]–[10]. Most of those protocols are developed under the assumption of benign environments, which makes them vulnerable to fail under various malicious attacks, e.g., denial-of-service (DoS) [17], message-delay attack [22] and message manipulation attack [24], etc.

Recently, the average-consensus-based time synchronization protocols have been proposed for achieving global time synchronization in wireless sensor networks [5], [6], [8]–[14]. The principle of average consensus is that each node takes an average of its own clock parameter and its neighboring ones to drive the network to achieve a consensus reference clock with an exponentially converging speed [9]. Since such mechanism does not rely on any specific reference node or network topology and can be implemented in a purely distributed and asynchronous way, it is inherently robust to certain kinds of malicious attacks, e.g., denial-of-service [17], node destruction [24], etc.

Unfortunately, the nature of average consensus makes the corresponding protocols vulnerable to a severe kind of security threat known as message manipulation, in which the external attackers or in-network compromised nodes inject corrupted messages into the network. Although the message manipulation attack is not new, its impact on consensus-based time synchronization protocol is devastating since such corrupted information will be propagated in an epidemic way. Therefore, the average consensus mechanism introduces new challenges in the detecting and possibly utilizing the manipulated messages as it permits all in-network nodes to actively utilizing the received messages.

In this paper, we focus on defending the average-consensus-based time synchronization (ATS) in wireless sensor network under message manipulation attacks. Different from most prior works which are mainly devoted to passively detecting and forbidding the malicious attacks [23]–[25], we propose a novel defense mechanism to allow the safe nodes to pro-actively accept and utilize the manipulated messages. The major contributions of this work are summarized as follows:

1) To the best of our knowledge, this is the first work that investigates the impact of message manipulation attack over ATS, and defends ATS against the message manipulation attack.
2) We analytically and numerically investigate the impact of message manipulation attack on the ATS protocol, and provide a necessary condition for ATS to converge under such attacks.
3) Both hardware clock checking process and adjusting parameter checking process are proposed to defend against the message manipulation for different protocol parameters. We further incorporate these processes into the original ATS to form a novel secure average-consensus-based time synchronization protocol (SATS).
4) We prove that SATS guarantees the network time synchronization with an exponentially converging speed.

This paper is organized as follows. Section II discusses related works. In Section III, the problem of secure time synchronization is formulated. Section IV analyzes the performance of ATS under message manipulation attacks while Section V

Manuscript received March 20, 2013; revised June 26, 2013 and September 19, 2013; accepted October 01, 2013. Date of publication October 17, 2013; date of current version November 15, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Huaiyu Dai. This work was supported in part by the Natural Science Foundation of China under Grants 61190113, 61222305, and the 863 High-Tech Project under Grant 2011AA040101-1. The work of L. Shi was supported by HK RGC GRF Grant 618612. *(Corresponding author: J. Chen.)*

J. He, P. Cheng, and J. Chen are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Zhejiang 310027, China (e-mail: jphe@iipc.zju.edu.cn; pcheng@iipc.zju.edu.cn; jmchen@iipc.zju.edu.cn).

L. Shi is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Kowloon 00852, Hong Kong (e-mail: eesling@ust.hk).

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/TSP.2013.2286102

presents the detailed SATS protocol. Simulation is presented in Section VI. Finally, Section VII concludes the paper.

II. Related Work

Secure time synchronization in WSNs has long been recognized as a difficult and important problem [18]–[20].

Many efforts have been devoted to providing secure time synchronization protocols in WSNs [21]–[28]. Most of these existing protocols achieve secure time synchronization by designing different safeguard techniques for transitional reference node-based and tree topology-based protocols, e.g., TPSN [3] and FTSP [4]. For example, Huang *et al.* propose several techniques to reinforce the structure of classical protocol FTSP to avoid attacks by malicious nodes [27]. Sun *et al.* propose a secure and resilient time synchronization subsystem (TinySeRSync) for WSNs running TinyOS, which uses authenticated technique to overcome attacks by malicious nodes [21]. By introducing high power nodes to form hierarchical topologies and using public-key based broadcast authentication, Du *et al.* achieve the secure time synchronization [31]. Benzaid *et al.* in [29] propose a secure pairwise broadcast time synchronization by using a public key cryptography based authentication scheme proposed in [30]. Song *et al.* develop two approaches to detect and accommodate a specific type of attack called delay attack, i.e., outlier-based and threshold-based techniques [22]. Using threshold-based technique, Ganeriwal *et al.* in [23] propose a suite of secure pairwise and group-wise synchronization protocol, which prevents the pulse-delay attack by checking if the end-to-end delays exceed a prescribed threshold. The authors in [25] tackle the man-in-the-middle attack, where the attacker could prevent the proper operation of the protocol, and present associated secure time synchronization protocol for WSNs. By using Pairing and Identity-based cryptography, a secure time synchronization protocol is proposed by Rahman *et al.* to reduce the communication and storage requirements for heterogenous sensor networks [28].

Most of existing works, however, only consider the compensation of clock offset, and rely on certain reference clocks and hierarchical structure. Therefore, they are more vulnerable to single node failure and intelligent attacks. Recently, Hu *et al.* propose a distributed and secure synchronization protocol, i.e., attack-tolerant time synchronization protocol (ATSP), which is more desirable for WSNs [24]. Although ATSP is able to accurately detect attacks and iteratively achieve synchronization across the network in a fully distributed manner, the clock skew errors are not compensated. Moreover, ATSP only promises bounded ultimate synchronization error.

Compared with the aforementioned protocols, our proposed SATS protocol has several advantages. First, SATS compensates both skew and offset, and is a fully distributed, asynchronous time synchronization protocol. Second, instead of identifying and prohibiting attack nodes, SATS pro-actively utilizes the corrupted messages from attack nodes. Third, we prove that SATS also has an exponential convergence speed, which is comparable with traditional ATS protocol.

III. System Model and Problem Setup

Consider a sensor network with $n(n > 3)$ safe nodes and $m(m \geq 1)$ malicious nodes (attack nodes), where the attack nodes can be external attackers or in-network nodes compromised by attackers. These $n + m$ nodes are assumed to have different and unique identity numbers, e.g., the safe and attack nodes are indexed by $1, 2, \ldots, n$ and $n+1, n+2, \ldots, n+m$, respectively. Assume each node only knows whether itself is an attack node without any pre-knowledge about other nodes.

TABLE I
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;
$x_i(t)$	the logical clock skew of safe node i at time t;
$y_i(t)$	the logical clock offset of safe node i at time t;
$\{\cdot\}_j^e$	the information of an attack node j;
$\mathcal{N}_i$	the neighbor node set of node i;
a_{ij}	the relative clock skew;
$\hat{a}_i$	the adjusting parameter for logical clock skew compensation;
$\hat{b}_i$	the adjusting parameter for logical clock offset compensation;
t_i	indicates a time for node i broadcast;
t_i^j	indicates a time that node i receives message from node j;
t^+	indicates the time just after updating at time t;
$\mathbf{N}^+$	the set of positive integers.

Let $\mathcal{G}(t) = (\mathcal{V}, \mathcal{E}(t))$ be the graph of whole network, where $\mathcal{V}$ denotes the set of vertices 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. Clearly, $\mathcal{V} = \mathcal{V}_s \cup \mathcal{V}_a$. For easing the later presentation, we assume that the graph composed by safe nodes is always connected. Consider the scenario that the attack nodes in the network are sparse, and any two attack nodes are thus assumed to be never neighboring with each other. We assume that the communication between nodes is reliable, and each node will send authenticated message so that the receiving node can only read or copy the message without any modification [19], [21], [23], [30]. Some important notation definitions are given in Table I.

A. Clock Model

Different from [3] where only clock offset is considered, we consider a more general hardware clock model as in [4], [16], [24]. Specifically, the hardware clock reader $\tau_i(t)$ of any node at time t is modeled as the following linear function

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

where a_i is the hardware clock skew which determines the clock speed and b_i is the hardware clock offset. As proved in [8], the hardware clock skews may be slightly different for each node due to several reasons, e.g., imperfect crystal oscillators, ambient temperature or battery voltage and oscillator aging. However, they can still be modeled as bounded values [8], i.e.,

$$1 - \varrho \leq a_i \leq 1 + \varrho, \ i \in \mathcal{V}, \tag{2}$$

where $0 \leq \varrho < 1$ is a constant.

It has been pointed out that a_i and b_i cannot be exactly calculated [9]. However, by comparing the local clock readings, the hardware clock of node i can be expressed as follows

$$\tau_i(t) = \frac{a_i}{a_j}\tau_j(t) + \left(b_i - \frac{a_i}{a_j}b_j\right) = a_{ji}\tau_j(t) + b_{ji}, \tag{3}$$

where $a_{ji} = \frac{a_i}{a_j}$ is the relative hardware clock skew 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[9].

The value of the hardware clock should not be adjusted manually as the other hardware components may depend on a continuously running hardware clock [8]. Hence, a logical clock value $L_i(t)$ is developed to represent the synchronized time of node i. It is calculated as a function of the current hardware clock, which is given by

$$L_i(t) = \hat{a}_i\tau_i(t) + \hat{b}_i = \hat{a}_i a_i t + \hat{a}_i b_i + \hat{b}_i, \tag{4}$$

where $\hat{a}_i a_i = x_i$ and $\hat{a}_i b_i + \hat{b}_i = y_i$ denote the logical clock skew and the logical clock offset, respectively.

The communication delay is also assumed to be ignored in this paper, which is the same as [8], [9], since we focus on the secure average consensus-based time synchronization. Note that Schenato *et al.* [9] pointed out that by utilizing the MAC layer time-stamping [32] available in many sensor network devices, the reading of the local clock $\tau_i(t_1)$ at the transmitting node and the reading of the local clock $\tau_j(t_2)$ at the receiving node are instantaneous, i.e., $t_1 = t_2$ (see Section V.C in [8] for a detailed description), thus the communication delay can be safely assumed to be 0. If this is not the case, the average consensus-based time synchronization protocols need to be modified to handle the packet delivery delay (see, e.g., [1], [6], [33], [34] for transmission delay compensation). A brief discussion on the effect of communication delay is provided in Section V-F.

B. Attack Model

Time synchronization protocols in WSNs are vulnerable to different security attacks including replay attack, sybil attack, message manipulation attack, delay attack and Dos attack, etc. [18], [23], [28]. In this paper, we mainly focus on the following attack.

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 [24], [38], [39].

From the above definition of Message manipulation, it follows that the replay attack and delay 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 regarded as adding a delay to the real message. In this work, we assume that the attack node has the ability to freely manipulate and broadcast its own hardware clock readings as well as all the adjusting parameters required in the ATS protocol.

It should be pointed out that there are some methods of signal processing, e.g., belief propagation [34], maximum likelihood estimation [35], [36], etc., which can be utilized to cope with time synchronization when the attack strategies of message manipulation taken by attackers are to add some random variables which follow certain distributions, such as Gaussian or exponential distribution, which can be viewed as delay attacks. For example, Kim *et al.* [33] combine Gaussian mixture Kalman particle filter with an iterative noise density estimation procedure to achieve robust time synchronization in the presence of unknown network delay distributions. And, under exponential delays, the maximum likelihood estimation proposed in [35], [36] can handle both clock skew and offset compensation. In this paper, the attackers can freely manipulate their message, which renders the signal processing methods proposed in [33]–[36] inapplicable.

C. Problem Setup

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

$$L_i(t) = \hat{a}_i\tau_i(t) + \hat{b}_i = \tau_v(t),\ i \in \mathcal{V}, \tag{5}$$

where $\tau_v(t)$ is a common clock, which is given by

$$\tau_v(t) = a_v t + b_v. \tag{6}$$

Thus, the goal of traditional time synchronization protocol is to find the adjusting parameters $\hat{a}_i$ and $\hat{b}_i$ for $\forall i \in \mathcal{V}$ such that (5) holds, i.e., 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, we will also have to take into account that the attack nodes aim to destroy the time synchronization as much as possible. Therefore, our goal is to design a clock synchronization protocol to update the local adjusting parameters $(\hat{a}_i(k), \hat{b}_i(k))$ for node $i \in \mathcal{V}_s$, where k is the local updating times, such that

$$\lim_{k\to\infty} \hat{a}_i(k)a_i = a_v, \tag{7}$$

$$\lim_{k\to\infty} \hat{a}_i(k)b_i + \hat{b}_i(k) = b_v. \tag{8}$$

IV. Understanding ATS Under Attacks

A. ATS Protocol

ATS is a typical average consensus-based time synchronization protocol, which is proposed in [9]. The key idea of ATS is to let each node update its adjusting parameters $\hat{a}_i$ and $\hat{b}_i$ according to the messages received from its neighbor nodes. The updates of $\hat{a}_i$ and $\hat{b}_i$ are respectively used to compensate the logical clock skew and offset of node i. Due to the space limitation, we only present the detailed description of skew compensation, and the offset compensation can be implemented in a similar way.

For ATS, each node i periodically broadcasts its clock information, including the hardware clock reading $\tau_i(t)$ and two adjusting parameters $\hat{a}_i(t)$ and $\hat{b}_i(t)$, to its neighbor nodes with a common period T, where $\frac{\tau_i(t)}{T} \in \mathbf{N}^+$. As soon as node i receives a packet from node j at time t, it will update $\hat{a}_i$ as follows

$$\hat{a}_i(t^+) = \rho_v\hat{a}_i(t) + (1-\rho_v)a_{ij}(t^+)\hat{a}_j(t), \tag{9}$$

where $\rho_v \in (0,1)$ is a given constant and $a_{ij}(t^+)$ is the relative skew estimation of node i with regard to node j at time t, and estimating the relative skew is given as

$$a_{ij}(t_1) = \frac{\tau_j(t_1) - \tau_j(t_0)}{\tau_i(t_1) - \tau_i(t_0)}, i, j \in \mathcal{V}, \tag{10}$$

where $(\tau_i(t_1), \tau_j(t_1))$ and $(\tau_i(t_0), \tau_j(t_0))$ are two pairs of hardware clock readings of node i and j at time instances $t_1 > t_0$. It follows from (10) that a_{ij} is a constant and satisfies $a_{ij} = \frac{a_j}{a_i}$. In summary, once node i receives time message $\tau_j(t_0)$ from node j, its current clock is recorded and temporally stored as $(\tau_i(t_0), \tau_j(t_0))$. Clearly, if node i receives the time message $\tau_j(t_1)$ from node j for the second time, the relative skew a_{ij} is obtained from (10) directly. After obtaining the relative skew a_{ji}, the relative hardware clock offset b_{ji} is obtained from (3), i.e., $b_{ji} = \tau_i(t) - a_{ji}\tau_j(t)$.

B. ATS Performance Under Attacks

Let node j be an attack node which broadcasts fake messages $\tau_j^e(t)$, $\hat{a}_j^e(t)$ and $\hat{b}_j^e(t)$, where the values of these fake messages can be arbitrarily chosen (e.g., constant data injection or random data injection) by node j. Following the standard procedures of ATS, if a safe node i receives such fake messages from node j, it will still estimate the relative skew as

$$a_{ij}^e(t_1) = \frac{\tau_j^e(t_1) - \tau_j^e(t_0)}{\tau_i(t_1) - \tau_i(t_0)} = a_{ij}(t_1)\frac{\tau_j^e(t_1) - \tau_j^e(t_0)}{\tau_j(t_1) - \tau_j(t_0)}, \quad (11)$$

where $a_{ij}(t_1)$ is the relative skew in (10). Since node j can freely set and send out its hardware clock $\tau_j^e(t)$, it can control $a_{ij}^e(t_1)$ without any constraint, e.g., if node j chooses $\tau_j^e(t)$ randomly, then $a_{ij}^e(t_1)$ is also a random number. Therefore, based on the fake messages, note i will update its adjusting parameter for logical clock skew as

$$\hat{a}_i(t^+) = \hat{a}_i(t) + (1-\rho_v)\left[a_{ij}^e(t)\hat{a}_j^e(t) - \hat{a}_i(t)\right]. \quad (12)$$

Let $x_i(t) = a_i\hat{a}_i(t)$ denote the logical clock skew of each safe node i, and $x = [x_1, x_2, \ldots, x_n]^T$, which denotes the logical clock skews of the entire network. By multiplying both sides of (9) and (12) by a_i, the updating process of the logical clock skew for node i can be expressed as

$$x_i(t^+) = \begin{cases} \rho_v x_i(t) + (1-\rho_v)x_j(t), j \in \mathcal{V}_s, \\ x_i(t) + \varepsilon_i(t), j \in \mathcal{V}_a, \end{cases} \quad (13)$$

where

$$\varepsilon_i(t) = (1-\rho_v)\left[x_j^e(t) - x_i(t)\right]$$

with $x_j^e(t) = a_i a_{ij}^e(t)\hat{a}_j^e(t)$. Thus, the updating process of all safe nodes can be described as

$$x(k+1) = P(k)x(k) + \varepsilon(k), \quad (14)$$

where k denotes the iterations, $P(k) \in \mathcal{R}^{n\times n}$ is a stochastic matrix with $P_{ii} = \rho_v$, $P_{ij} = 1-\rho_v$, $P_{ll} = 1(l \neq i)$ and all the other elements equal to 0 if node i is updated at iteration k, and $\varepsilon_i(k) = 0$ for each safe node i and $\varepsilon_j(k)$ can be an arbitrary real number for attack node j which depends on its attack strategy at time k. Clearly, the skew compensation can be finished if and only if $\varepsilon(k)$ satisfies the convergence conditions of the discrete-time system (14). Then, a necessary condition is provided to guarantee the convergence of the discrete-time system (14), which is given by the following theorem.

Theorem 1: Consider the discrete-time system (14), if

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

where c is a constant and $\mathbf{1} = [1, 1, \ldots, 1]^T$, then it is necessary that $\varepsilon(k)$ must satisfy

$$\lim_{k\to\infty} \varepsilon(k) = \mathbf{0}. \quad (15)$$

Proof: Taking limitation on both sides of (14), it yields

$$c\mathbf{1} = \lim_{k\to\infty} P(k)c\mathbf{1} + \lim_{k\to\infty} \varepsilon(k). \quad (16)$$

Since each $P(k)$ is a stochastic matrix, the matrix $\lim_{k\to\infty} P(k)$ is still a stochastic matrix, which means that $\lim_{k\to\infty} P(k)c\mathbf{1} = c\mathbf{1}$. Substituting the above equation into (16) leads to (15). The proof is thus completed. ■

It is inferred from Theorem 1 that if there exist attack nodes which are able to make (15) violated, the time synchronization cannot be achieved by ATS protocol. Note that without any secure add-on process for ATS, each attack node can violate (15) easily. For example, each attack node j can send out a manipulated logical clock skew $\hat{a}_j^e(t) = \hat{a}_i(t)a_{ji} + \epsilon_j(t)$, where $\epsilon_j(t)$ can be freely determined by node j. Specifically, if $\epsilon_j(t) = \epsilon \neq 0$, it is a constant data injection attack; if $\epsilon_j(t)$ is a random number, it is a random data injection attack. Then, it easily makes $\lim_{k\to\infty} \varepsilon(k) \neq \mathbf{0}$ or even makes this limitation not exist.

In order to show the performance of ATS under message manipulations, the following simulation is conducted on a ring network with 30 nodes. Node 10 is assumed to be an attack node in this network, whose attack strategy is that $\hat{a}_{10}^e = \hat{a}_{10} + \omega_{10}$, where ω_{10} is decided by the attack strategy. Let $d_s(t)$ be the maximum difference between the logical skews of any two safe nodes, i.e., $d_s(t) = \max_{i,j\in\mathcal{V}_s}\{\hat{a}_i(t)a_i - \hat{a}_j(t)a_j\}$. Obviously, if $d_s(t)$ tends to 0, skew compensation is finished.Fig. 1(a) shows the trajectories of $d_s(t)$ for the case that node 10 does not make any attack ($\omega_{10} = 0$) and makes random data injection attack (ω_{10} is randomly chosen from $[0, 0.01]$). It can be observed that d_s cannot converge to 0 anymore under the attack, and will increase with the increasing number of attacks. Note that such a deviation of clock skew would further seriously increase the clock error among the sensor nodes when the time increases. Based on the results of Theorem 1, we further simulate the time synchronization performance of ATS under two different kinds of attack patterns, which is plotted in Fig. 1(b). Specifically, for strategy 1, the attack node randomly choose ω_{10} so that its logical clock skew always keeps between the logical clock skews of its two neighboring nodes, although it is still not a legal updating process. For strategy 2, the attack node remains to randomly select ω_{10} from $[0, 0.01]$. It is understood that for strategy 2, d_s still diverges. However, for strategy 1, it is interesting to observe that the time synchronization is finally achieved when the node's attack is bounded by its neighboring nodes' values consecutively, although the converging time is longer than that under no attacks.

Such an observation shows that it may not need to totally identify and cancel all illegal updating for time synchronization. Instead, we may be able to design certain protocols to constrain

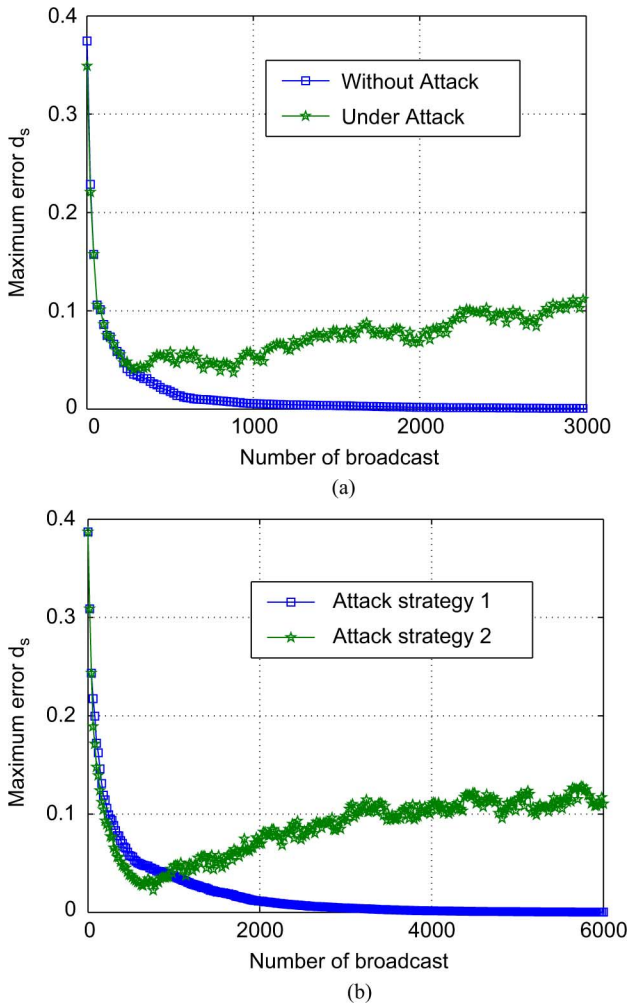

Fig. 1. The performance of ATS. (a) With and without attack. (b) Different attacks.

the attack pattern of malicious nodes so that the time synchronization can still be guaranteed.

C. Design Challenges

Based on the previous analysis and discussion, we summarize the major design challenges for a secure consensus-based time synchronization protocol as follows.

- **More interchanging parameters**: besides hardware clock reading τ_i, more parameters, e,g., $\hat{a}_i$ and $\hat{b}_i$ are required for ATS to compensate both clock skews and offsets. Hence, an attacker can freely choose one of them to attack, and the attackers have more opportunities to attack, which makes the problem more challenging.
- **No synchronized time information**: even if there is no concern about the limited resources, it is not an easy problem. For example, if the protocol requires, node i records all its neighbors' clock information in previous updating period and also sends them together to node j, then node j may not be able to check whether node i's updating is exactly legal as node j has no true information about the order by which node i received all neighboring information.
- **Limited resources**: each wireless sensor node usually has very limited resources, e.g., communication bandwidth, storage and energy, which makes the design even more challenging.

Despite the design challenges, we also observe the intuitions for designing a secure consensus-based time synchronization protocols from our previous analysis. Specifically, from Theorem 1 and the simulation results shown in Fig. 1(b), we find that it may be unnecessary to keep all parameters' updating exactly the same as the average consensus algorithm. Instead, if we can control the attack patterns, which corresponds to the $\varepsilon(k)$ in (15), we may still be able to guarantee the convergence of time synchronization throughout the network. In the following section, we will explicitly present our lightweight design based on such intuitions, and prove the convergence.

V. Main Design of SATS

Since the updating process of ATS mainly includes two kinds of parameters, the hardware clock reading τ_i and the adjusting parameters for logical clock, i.e., a_i for skew and b_i for offset, our SATS will basically consist of two checking processes which aim to guarantee that the provided parameters from the neighboring nodes are informative enough for the convergence of ATS algorithm.

A. Pseudo-Periodic Broadcast

To simplify the statements, we assume that each node i periodically transmits a packet to all of its neighbor nodes under a common period T based on its own hardware clock, i.e., the broadcast instants $t_i(k)$ are defined as $\tau_i(t_i(k)) = kT$ or equivalently $t_i(k) = \frac{kT-b_i}{a_i} = kT_i - \frac{b_i}{a_i}$. Note that the real broadcasting period for node i is $T_i = \frac{T}{a_i}$, which will be usually different for each node due to the difference of a_i. Therefore, such a broadcasting mechanism is also referred as pseudo-periodic broadcast.

Since a_i satisfies (2), we have $\frac{T}{1+\varrho} \leq T_i \leq \frac{T}{1-\varrho}$, which is valid for any node. Hence, if node i has received the message from its neighbor node j at its hardware time $\tau_i(t_0)$, it can be guaranteed that node i will receive the next message from node j within $\tilde{T} = \frac{1+\varrho}{1-\varrho}T$. Such a bounded time period, makes it possible for each node to collect the information from all its neighboring nodes within a constant duration based on its own hardware clock. It should be pointed out that the bounded time $\tilde{T}$ is important for later logical clock checking, as it is used for preventing the attack nodes using out-of-date receiving information to cheat.

B. Hardware Clock Checking Process

Since the hardware clock is linear, it is not difficult to check whether the neighboring node has sent a legal local hardware clock reading based on the historical messages. Specifically, let $s_{ij}(k)$ be the one-step relative skew estimation of node i for node j at the $k+1$-th receiving round, where $s_{ij}(k)$ can be calculated by

$$s_{ij}(k) = \frac{\tau_j(t_k) - \tau_j(t_{k-1})}{\tau_i(t_k) - \tau_i(t_{k-1})}. \tag{17}$$

Then, by comparing the two consecutive estimated relative hardware skews $s_{ij}(k)$ and $s_{ij}(k-1)$, node i can determine whether the hardware clock reading from node j is legal or not, i.e., the hardware clock reading is legal if and only if

$$s_{ij}(k) = s_{ij}(k-1), \tag{18}$$

and therefore drop the illegal messages. Based on such hardware clock checking process, it can be guaranteed that all accepted hardware clock readings of each node (even the attack nodes) will remain as a linear function of the real time. Such mechanism guarantees that the estimated relative skews, a_{ij}, remain constant, which means the attack nodes will have no chance to inject corrupted hardware clock readings into the messages.

C. Logical Clock Checking Process

Different from the hardware clock readings, there is no simple changing pattern for the adjusting parameters for the logical clock, which makes it intrinsically difficult to prevent manipulation completely. However, as shown in Section IV, we may still be able to achieve time synchronization as long as the received adjusting parameters are informative enough which means they are properly restrained.

In this part, we first propose a checking process for the logical clock updating so that the attack node can only send out its own adjusting parameters with dynamic upper and lower bounds, where the dynamic upper and lower bounds depend on the larger and smaller logical clocks among its neighboring nodes, respectively. Otherwise the parameters will never be accepted by any safe node. And, it will be easy for the safe nodes to satisfy such a checking process due to the characteristic of average consensus. With such properties, in later part, we prove that the overall SATS guarantees the convergence of all logical clocks throughout the network.

Let $\Theta_i^j(t)$ be the information set of node i that it needs to send to its neighbor node j, which is given by

$$\Theta_i^j(t) = \{i, \tau_i(t), \hat{a}_i(t), \hat{b}_i(t), a_{ij}, (\tau_i(t_i^j), \tau_j(t_i^j))\},$$

where i is the ID number, τ_i is the hardware clock reading, $\hat{a}_i$ and $\hat{b}_i$ are adjusting parameters, a_{ij} is the relative skews, and $(\tau_i(t_i^j), \tau_j(t_i^j))$ is one historical pair of hardware clock reading which is stored in node i. Furthermore, let $\Theta_i(t)$ be the common information set of an arbitrary node i, which consists of all $\Theta_i^j(t)$ for $j \in \mathcal{N}_i$, i.e.,

$$\Theta_i(t) = \cup_{j\in\mathcal{N}_i}\Theta_i^j(t).$$

After the hardware clock checking process is finished, each node i has obtained a_{ij} for each creditable neighbor node j, and the associated $(\tau_i(t_i^j), \tau_j(t_i^j))$ is also stored in the node i, where t_i^j is the time at which the node i receives the latest message from node j. Clearly, once successfully accepting credible messages from its neighboring nodes, node i will instantly update its own local information set.

Now, we introduce the message content that each node will broadcast, which mainly consists of three parts. For each node i, except its own common information set $\Theta_i(t_i)$, the message content also includes $\Theta_{i_0}^i(t_{i_0})$ and $\Theta_{i_1}^i(t_{i_1})$, which are selected from the received common information sets $\Theta_{i_0}(t_{i_0})$ and $\Theta_{i_1}(t_{i_1})$. Note that different from traditional ATS protocol, besides its own clock information, it further requires the clock information of its two neighboring nodes. Specifically, $\Theta_{i_0}^i(t_{i_0})$ is the clock information of the neighbor node with the smaller logical clock skew than node i while $\Theta_{i_1}^i(t_{i_1})$ is the clock information of the one with the larger logical clock skew than node i (If all the logical skews are the same then we compare the logical offsets). For statement clarification, for the receiving node i, we simply denote i_0 as the node with the smallest logical clock skew, and i_1 as the node with the largest one[1]. Therefore, once successfully accepting credible message from a neighbor node j, node i will update $\Theta_{i_0}(t_{i_0})$ and $\Theta_{i_1}(t_{i_1})$ based on $\Theta_j(t_j)$, such that i_0 is always the node with the smallest logical clock skew, and i_1 is always the node with the largest one. For example, if node i successfully accepts credible messages $\Theta_j(t_j)$ and the logical clock skew of node j is less than its current i_0's logical clock skew, then the node j will replace current i_0, and then $\Theta_{i_0}(t_{i_0})$ is changed as $\Theta_j^i(t_j)$.

It follows from the above discussion that the information for each node i broadcasting includes $\Theta_i(t_i)$, $\Theta_{i_0}^i(t_{i_0})$ and $\Theta_{i_1}^i(t_{i_1})$. Hence, one can infer that the information stored at each node i includes the relative skew a_{ij}, one historical pair of hardware clock reading $(\tau_i(t_i^j), \tau_j(t_i^j))$ for each neighbor $j, j \neq i_0, i_1$, and the information sets $\Theta_{i_0}^i(t_{i_0})$ and $\Theta_{i_1}^i(t_{i_1})$ for neighbor nodes i_0 and i_1, respectively.

Then, for the adjusting parameter checking process, each safe node will directly check whether the received message contains such three parts of information. For example, if node j receives a message from node i, then in such message, node i must provide the original information of its two neighboring nodes, which have larger and smaller logical skews than node i itself. Otherwise, node j will not accept the message from node i. The detailed logical clock checking process is given as follows.

• **Logical clock checking process**: suppose that node j receives $\Theta_i(t_i)$, $\Theta_{i_0}^i(t_{i_0})$ and $\Theta_{i_1}^i(t_{i_1})$, from node i, then node j checks the following four conditions,

c_1. $i \neq i_0 \neq i_1$.

c_2. $\frac{\tau_i(t_i^\nu)-\tau_i(t_i^i)}{\tau_\nu(t_\nu)-\tau_\nu(t_\nu^i)} = a_{\nu i}$ and $\tau_i(t_i) - \tau_i(t_i^\nu) \leq \tilde{T}$, where $\nu = i_0, i_1$.

c_3. $\frac{\hat{a}_{i_0}(t_{i_0})}{\hat{a}_i(t_i)a_{i_0 i}} \leq 1 \leq \frac{\hat{a}_{i_1}(t_{i_1})}{\hat{a}_i(t_i)a_{i_1 i}}$.

c_4. $\phi_{ii_0}(t_{i_0}) \leq 0 \leq \phi_{ii_1}(t_{i_1})$, where $\phi_{i\nu}(t_\nu)$ is defined as

$$\phi_{i\nu}(t_\nu) = \hat{a}_\nu(t_\nu)\tau_\nu(t_\nu^i) + \hat{b}_\nu(t_\nu) - \hat{a}_i(t_i)\tau_i(t_\nu^i) - \hat{b}_i(t_i), \tag{19}$$

where $\nu = i_0, i_1$.

If conditions c_1, c_2 and c_3 hold, then the logical clock updating parameter $\hat{a}_i(t_i)$ is credible for node j; if conditions c_1, c_2 and c_4 hold, the logical clock offset updating parameter $\hat{b}_i(t_i)$ is credible for the node j.

Lemma 1: If the conditions c_1 and c_2 hold, then $\Theta_{i_0}^i(t_{i_0})$ and $\Theta_{i_1}^i(t_{i_1})$ for node i broadcast are received from two different nodes i_0 and i_1, and both of them are received in the time interval $[t_i - \frac{\tilde{T}}{(1-\varrho)}, t_i]$, i.e.,

$$t_i - \frac{\tilde{T}}{(1-\varrho)} \leq t_\nu \leq t_i, \nu = i_0, i_1. \tag{20}$$

Proof: The proof is provided in the Appendix A. ■

[1]Node i is allowed to select another pair of nodes with larger and lower logical skew, respectively, which will not affect our convergence proof.

In light of this result, this process guarantees that node i's logical clock is bounded by its two neighbor nodes. Specifically,

1) The $\hat{a}_i(t_i)$ is credible for the node j if it guarantees that the logical clock skew of i at time t_i is bounded by the logical clock skews of two different neighbor nodes nodes i_0 and i_1, i.e., $x_{i_0}(t_{i_0}) \le x_i(t_i) \le x_{i_1}(t_{i_1})$, where t_i, t_{i_0} and t_{i_1} satisfy (20).
2) If $x_\nu(t_\nu) = x_i(t_i)$ holds for $\nu = i_0, i_1$, then the parameter $\hat{b}_i(t_i)$ is credible for the node j when it guarantees that the logical clock offset of i at time t_i is bounded by the logical clock offsets of two different neighbor nodes nodes i_0 and i_1, i.e., $y_{i_0}(t_{i_0}) \le y_i(t_i) \le y_{i_1}(t_{i_1})$, where t_i, t_{i_0} and t_{i_1} satisfy (20).

There are two key points within the logical clock checking process. One is that node j uses the information sets $\Theta^i_{i_0}(t_{i_0})$ and $\Theta^i_{i_1}(t_{i_1})$ of two different nodes i_0 and i_1 which cannot be manipulated by node i, to check the information set $\Theta_i(t_i)$ created by node i. The other is that the information $\Theta^i_{i_0}(t_{i_0})$ and $\Theta^i_{i_1}(t_{i_1})$ transmitted by node i should be lately created by i_0 and i_1 in the time interval $[t_i - \frac{\tilde{T}}{(1-\varrho)}, t_i]$, respectively, which is guaranteed by Lemma 1.

Note that Yan *et al.* in [38] considered the security problem of consensus-based spectrum sensing in cognitive ratio networks. The authors proposed a promising distributed hash-based verification mechanism to ensure that each neighbor node performs trustworthy state update. The proposed mechanism uses the convergence property of the consensus algorithm to bound the malicious data and utilizes two-hop neighbor nodes date to make the verification, which is the same as SATS. However, it could not be used to solve the secure time synchronization problem considered in this paper directly for the following two reasons. First, the mechanism proposed in [38] is based on synchronous average consensus, while ATS is based on an asynchronous consensus where nodes have no access to synchronized time information. Second, since different nodes are with different hardware clocks, each safe node needs to verify whether the information of two-hop neighbors are created in the latest time to avoid being attacked by the attackers using earlier neighbors' information.

D. SATS Protocol

We describe the detailed SATS protocol in Algorithm 1, which is based on the average consensus concept with the combination of our proposed hardware clock checking and logical clock checking processes.

Algorithm 1 *Secure Average Time Synchronization (SATS)*

Initialization:

1) Let the initial adjusting parameters of each node i be $\hat{a}_i(0) = 1$ and $\hat{b}_i(0) = 0$ for $i \in \mathcal{V}$, and give the common broadcast period T to each node.

Iteration:

2) If the current hardware clock reading $\tau_i(t_i)$ of node i satisfies $\frac{\tau_i(t_i)}{T} \in \mathbf{N}^+$, then go to next step.
3) Compute $\frac{\hat{a}_{i_0}(t_{i_0})}{\hat{a}_i(t_i)} a_{ii_0}$ and $\frac{\hat{a}_{i_1}(t_{i_1})}{\hat{a}_i(t_i)} a_{ii_1}$, if $\frac{\hat{a}_{i_0}(t_{i_0})}{\hat{a}_i(t_i)} a_{ii_0} > 1$,

$$\hat{a}_i \leftarrow \hat{a}_{i_0}(t_{i_0}) a_{ii_0}; \tag{21}$$

and if $\frac{\hat{a}_{i_0}(t_{i_1})}{\hat{a}_i(t_i)} a_{ii_1} < 1$,

$$\hat{a}_i \leftarrow \hat{a}_{i_1}(t_{i_1}) a_{ii_1}. \tag{22}$$

4) Compute $\phi_{ii_0}(t_{i_0})$ and $\phi_{ii_1}(t_{i_1})$ by**(19)**, if $\phi_{ii_0}(t_{i_0}) > 0$,

$$\hat{b}_i \leftarrow \hat{a}_{i_0} \tau_{i_0}(t_{i_0}) + \hat{b}_{i_0} - \hat{a}_i \tau_i(t_{i_0});$$

and if $\phi_{ii_1}(t_{i_1}) < 0$,

$$\hat{b}_i \leftarrow \hat{a}_{i_1} \tau_{i_1}(t_{i_1}) + \hat{b}_{i_1} - \hat{a}_i \tau_i(t_{i_1}).$$

5) Create the common information set $\Theta_i(t_i)$, and then broadcast a packet, including $\Theta_i(t_i)$, $\Theta^i_{i_0}(t_{i_0})$ and $\Theta^i_{i_1}(t_{i_1})$, to the neighbor nodes.

Checking after Receiving a Packet:

6) If node j receives a packet from node i at time t_i, it records its local hardware clock reading $\tau_j(t_i)$ and builds a pair of hardware clock readings $(\tau_j(t^i_j), \tau_i(t^i_j)) = (\tau_j(t_i), \tau_i(t_i))$.
7) Do hardware clock reading checking process: if the hardware clock reading is legal, delete earlier stored message $(\tau_j(t), \tau_i(t))$ for $\forall t < t_i$ and go to the next step.
8) Do logical clock checking process: if c_1, c_2 and c_3 hold, then $\hat{a}_i$ is legal; If c_1, c_2 and c_4 hold, then $\hat{b}_i$ is legal.

Average Consensus:

9) If $\hat{a}_i$ is legal, then

$$\hat{a}_j \leftarrow \rho_v \hat{a}_j + (1 - \rho_v) a_{ji} \hat{a}_i. \tag{23}$$

If $\hat{b}_i$ is legal, then

$$\hat{b}_j \leftarrow \hat{b}_j + (1 - \rho_o) \left[\hat{a}_i \tau_i(t_i) + \hat{b}_i - \left(\hat{a}_j \tau_j(t_i) + \hat{b}_j \right) \right].$$

10) Update $\Theta^j_{j_0}(t_{j_0})$ and $\Theta^j_{j_1}(t_{j_1})$ based on the received $\Theta_i(t_i)$.

In step 3 and step 4, before node i intends to broadcast, it will update its logical clock (include logical skew and offset) when it has the largest or smallest logical clock larger among all the neighbor nodes, such that it has the same logical clock as i_1 or i_0. These two steps guarantee that the logical clocks of safe nodes are always bounded by two neighbor nodes, and thus can pass the logical clock checking process.

E. Convergence Analysis of SATS

In this part, we analyze the convergence of SATS. The detailed proofs will be provided in the Appendix B.

Theorem 2: Through the SATS protocol depicted in Algorithm 1, the logical clock skew through the network will converge to a constant value exponentially, i.e.,

$$\lim_{t \to \infty} x(t) = a_v \mathbf{1},$$

exponentially fast, where $a_v \in [\min_{i \in \mathcal{V}_s} a_i, \max_{i \in \mathcal{V}_s} a_i]$.

Similarly, for the logical clock offset compensation, we have the similar result. Therefore, the convergence of SATS is immediately obtained.

Theorem 3: Through the SATS protocol depicted in Algorithm 1, the difference between any two safe nodes will converge to 0 exponentially, i.e.,

$$\lim_{t\to\infty}\{L_i(t)-L_j(t)\}=0, i,j\in\mathcal{V}_s,$$

exponentially fast for $\forall i,j\in\mathcal{V}_s$.

From the above theorem, it follows that by SATS the logical clocks of safe nodes will exponentially converge, which has the same convergence speed as the traditional ATS proposed in [9]. It also note that in order to guarantee all the received messages are informative even under malicious message manipulation attacks, each safe node will have to maintain a larger local information sets $\Theta_i(t_i)$, $\Theta^i_{i_0}(t_{i_0})$ and $\Theta^i_{i_1}(t_{i_1})$, and broadcast them to the neighbor nodes, which increases the storing and communication costs. In other words, security does come with a price.

F. Communication Delay and Attack Cooperation

In this subsection, we will discuss the performance of SATS with the consideration of the communication delay and attack cooperation, respectively. Taking communication delay into consideration, the hardware clock readings $\tau_j(t_k), k\in 0\cup\mathbf{N}^+$, for each node i received from neighbor node j will satisfy

$$\tau_j(t_k)=a_jt_k+b_i-d(t_k),$$

where $d(t_k)$ denotes the communication delay which satisfies $\frac{d}{2}>d(t_k)>0$, and $\frac{d}{2}$ is the upper bound of the communication delay. Then, each one-step relative skew estimation $s_{ij}(k)$ based on (17) will fluctuate and not equal to a constant. Hence

$$\begin{aligned}s_{ij}(k)&=\frac{a_j(t_k-t_{k-1})-d(t_k)+d(t_{k-1})}{a_i(t_k-t_{k-1})}\\&=\frac{a_j}{a_i}-\frac{d(t_k)-d(t_{k-1})}{a_i(t_k-t_{k-1})}\end{aligned}\tag{24}$$

holds for $\forall j\in\mathcal{N}_i$. It will render (18) false, i.e., none of hardware clock readings can pass the hardware clock checking process. Therefore, (18) is replaced by

$$\begin{aligned}&|s_{ij}(k)-s_{ij}(1)|\\&\quad=\left|\frac{d(t_k)-d(t_{k-1})}{a_i(t_k-t_{k-1})}-\frac{d(t_1)-d(t_0)}{a_i(t_1-t_0)}\right|\\&\quad\le\frac{d}{(1-\varrho)T},\ \forall j\in\mathcal{N}_i.\end{aligned}\tag{25}$$

According to (25), it guarantees that the hardware clock readings of all the safe nodes can still pass the hardware clock checking process when communication delay is considered. Meanwhile, under the constraint of (25), each one-step relative skew estimation $s_{ij}(k)$ fluctuates with a constant $s_{ij}(1)$, and the fluctuation is bounded by $\frac{d}{(1-\varrho)T}$. It follows that the hardware clock readings received from neighbor nodes are approximately linear functions, which can restrain the attack nodes from freely modifying hardware clock reading to attack. Moreover, by setting $T\gg d$ such that the fluctuation bound $\frac{d}{(1-\varrho)T}$ is small enough, we can further counteract the effect of communication delay and can restrain the attack nodes better[2]. Similarly, for the logical clock checking process, to ensure that the safe nodes can pass this process, we need to revise the condition c_2 as follows

$$c_2'\cdot\left|\frac{\tau_i(t_i^\nu)-\tau_i(t_\nu^i)}{\tau_\nu(t_\nu)-\tau_\nu(t_\nu^i)}-a_{\nu i}\right|\le\frac{d}{(1-\varrho)t}\ \text{and}\ \tau_i(t_i)-\tau_i(t_i^\nu)\le\tilde{T}$$

where $\nu=i_0,i_1$.

By similar analysis of condition c_2 in Lemma 1, we can infer that condition c_2' also ensures that the information $\Theta^i_{i_0}(t_{i_0})$ and $\Theta^i_{i_1}(t_{i_1})$ are latest received by node i from i_0 and i_1 in the time interval $[t_i-\frac{\tilde{T}}{(1-\varrho)}-d,t_i]$, respectively. After the above revisions for hardware and logical clock checking processes, none of the safe nodes will be viewed as attack nodes and isolated by their neighbor nodes, while to avoid being detected by the safe nodes, the hardware clock readings for attacker broadcast should still be approximately linear functions of real time t. Hence, by SATS, the time synchronization can also be achieved when communication delay exists. However, the synchronization accuracy is affected by the communication delay, and the detailed analysis will be given in the simulation section.

Note that for SATS, each node can pass the logical checking process only when its logical clock is not less than the smallest logical clock or greater than the largest logical clock among its neighbor nodes (not include its own logical clock). That is, by utilizing the logical checking process, each attack node should maintain its logical clock between its neighbor nodes' logical clocks to avoid being detected and isolated by safe neighbor nodes. If the attack nodes cannot collude with each other, i.e., each attack node cannot utilize the attack information of any other attacker to attack, every attack node should ensure that the message of itself for broadcast can pass both the hardware and logical clock process. Otherwise, the attack node which is neighboring with the safe node will be detected and isolated by the safe neighbor nodes, i.e., all the safe nodes will not use the attack nodes' information for clock updating. And, if the clock information of every node can pass the hardware checking process and logical checking process, the time synchronization of all safe nodes can be achieved. Thus, the assumption that two attack nodes are never neighboring with each other can be relaxed to that the attackers cannot collude with each other to attack. Many existing works on secure time synchronization, e.g., [21]–[26], assumed that attackers are not able to collude. If the attackers can cooperate with each other, one attack node can always broadcast incorrect clock information, i.e., $\Theta^i_{i_0}(t_{i_0})$ or $\Theta^i_{i_1}(t_{i_1})$), to its neighboring attack node i, and help node i with illegal adjust parameters to pass the logical checking process, so that the convergence of SATS cannot be guaranteed. Since each attack node does not know the identity (attack or safe) of the other nodes, it should inform its neighboring attack nodes of its attack identity before cooperation. For SATS, each attack node can inform its attack neighbor nodes of its attack identity by broadcasting incorrect clock information such that this information cannot pass the checking processes of these neighbor

[2]A larger T will decrease the convergence speed of the algorithm, which renders a tradeoff. This is another interesting problem, and will be left as our future work.

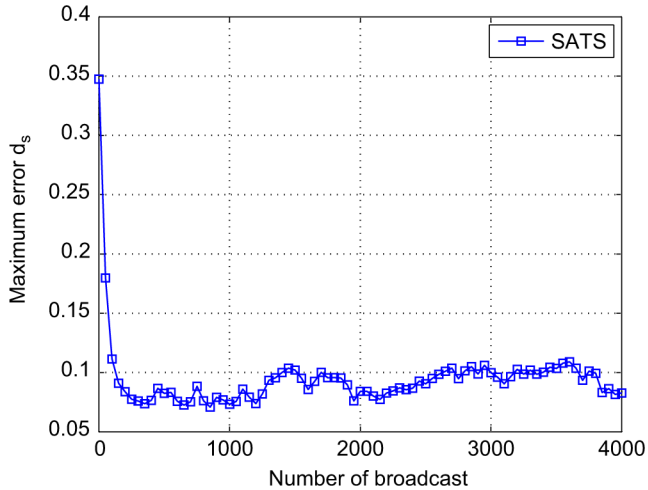

Fig. 2. The performance of SATS in a ring network under cooperation attacks.

nodes. However, it is difficult to be realized without being detected by the neighboring safe nodes, except all its neighboring nodes are attack nodes. If there are attack nodes not neighboring with any safe nodes, the problem will become much more challenging as the attack nodes can realize cooperation attack and disrupt time synchronization (see the following example as illustration), which is beyond the scope of the current paper and will be pursued in our future research.

Example 1: Consider the same network scenario as in Fig. 1. Assume that attack node 10's neighbor nodes 9 and 11 are also attackers and $a_{10} = \max_{i\in\mathcal{V}} a_i$, i.e., attack node 10 is not neighboring with any safe nodes and with maximum hardware clock skew (i.e., with maximum initial logical clock skew). The attack strategy for node 10 is that $\hat{a}^e_{10} = \hat{a}_{10} + \omega_{10}$, where ω_{10} is randomly selected from $[0, 0.01]$. According to SATS, although nodes 9 and 11 can easily obtain the attack identity of node 10 as its logical clock under the attack strategy cannot pass the logical clock checking process of them, they still use node 10's information for their clock updating and use as $\Theta^i_{i_1}(t_{i_1})$ $(i = 9, 10)$ for them, i.e., nodes 9 and 11 cooperate with 10 to attack. Then, as shown in Fig. 2, SATS cannot converge and the time synchronization cannot be achieved.

VI. Performance Evaluation

This section conducts extensive simulations to evaluate our design under different network settings as well as attack patterns, and some insights of the performance are revealed.

A. Simulation Setup

Throughout the simulation examples, we assume the wireless sensor nodes are deployed over a $100m * 100m$ two-dimensional square area. Except where otherwise specified, the default number of sensor nodes is 50 where the communication range of each node is set to be 30 meters. We set $\hat{a}(0) = 1, \hat{b}(0) = 0$ and $T = 1$, and for each node i, let the hardware clock skew a_i be randomly selected from the interval $[0.8, 1.2]$ and the hardware clock offset b_i from the interval $[0, 0.4]$. For the average consensus process in SATS, let $\rho_v = \rho_o = \frac{1}{2}$ as in [9]. Let each attack node m_i follow $\hat{a}^e_{m_i} = \hat{a}_{m_i} + \omega_{m_i}$ to manipulate its clock compensation parameters for $i = 1, 2, \ldots, m$. We mainly consider two typical kinds of manipulations, i.e., random data injection with ω_{m_i} randomly selected in $[0, 0.01]$, and constant data injection with $\omega_{m_i} = 0.01$. Since the clock offset compensation is implemented in a similar way as the skew compensation, we will mainly show the evaluation of the skew compensation. In order to show the synchronization accuracy, we define the maximum skew error among all safe nodes, i.e.,

$$V(x(t)) = \max_{i,j\in\mathcal{V}_s} \{x_i(t) - x_j(t)\}.$$

Note that even a small skew error $V(x)$ can lead to an increasingly larger logical clock difference as the time increases.

B. Sats vs Traditional ATS Under Attacks

In this part, we compare our SATS protocol with traditional ATS protocol [9] for various system settings.

First, consider the ring network with 30 nodes. If all nodes are safe nodes, i.e., there is no attack within the ring network, we compare the performance of SATS and ATS, which is shown in Fig. 3(a). It is observed that SATS converges at the same speed as ATS in the safe environment which is also supported by Theorem 2. Then, we consider the situation where node 10 initiates the random data injection attack from the very beginning, and compare the time synchronization performance of SATS and ATS in Fig. 3(b). It can be seen that the skew error does increase for ATS, making the unbounded logical clock error within the network, which has also been proved by the necessary condition provided in Theorem 1. On the other hand, our proposed SATS protocol still ensures the exponential convergence of clock skew error as proved by Theorem 2.

Furthermore, consider a random graph with $n = 50$ and $m = 4$, which means that there are 50 safe nodes and 4 malicious nodes in the network. Fig. 4(a) shows the performance of SATS and ATS when the attackers do not manipulate messages. Also note that SATS converges almost the same fast as ATS in this safe environment. The performance of SATS and ATS under random data injection attack is shown in Fig. 4(b), which again indicates that SATS is resilient under the message manipulation attacks.

C. Performance of Sats Under Different Attack Parameters

In this part, we show the performance of SATS under various number of attack numbers, attack strategies, and attack frequencies.

We first vary the number of attack nodes, and show the performance of SATS for both ring and random topologies, which are depicted in Fig. 5. Note that for comparison, we also add the performance under no attacks as the reference line. It can be observed that for both kinds of topologies with various attacker numbers, SATS guarantees that the network clock skew error converges exponentially. Another interesting observation is that with the increasing number of attack nodes, the converging speed appears to be even faster. The reason is that, instead of rejecting all messages from attack nodes, our SATS allows the safe nodes to accept the informative clock messages from attack nodes. Such a novel mechanism in fact utilizes the attack nodes to help increase the network connectivity and deliver informative clock information, therefore improving the convergence speed of the whole network. Moreover, when the

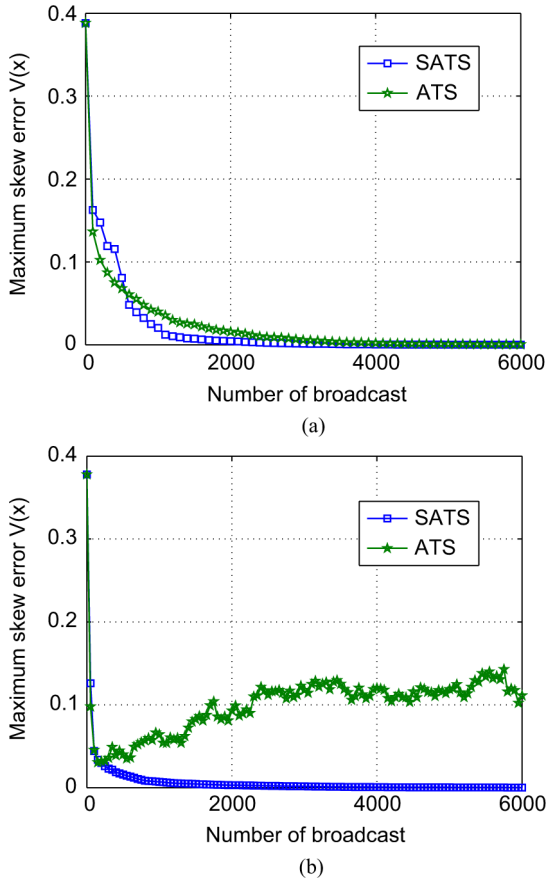

Fig. 3. The performance of ATS and SATS in a ring network (a) Without attacks (b) Random data injection attack.

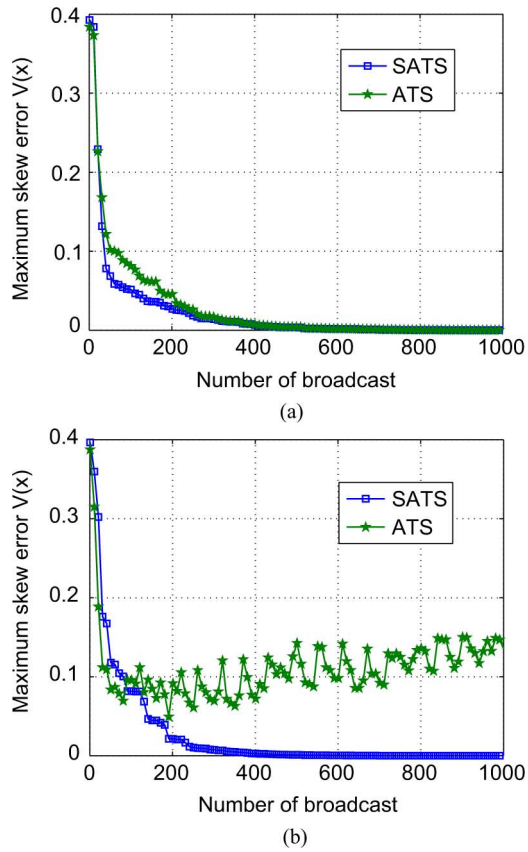

Fig. 4. The performance of ATS and SATS in a random network (a) Without attacks (b) Random data injection attack.

percentage of attackers is increasing, the convergence speed of SATS will not change notably. For example, based on 50 tests, in the random network, when the percentage of attackers are 0%, 10%, and 20%, the average number of broadcast of all safe nodes are respectively more than 853, 628 and 665 to ensure $V(x(t)) \leq 10^{-4}$, and 1493, 1311 and 1230 to ensure $V(x(t)) \leq 10^{-6}$.

We also compare the performance of SATS under different attack strategies, i.e., constant data injection and random data injection. As shown in Fig. 6(a), for both attack strategies, SATS achieves network time synchronization rapidly. Meanwhile, it can be seen that the convergence speed under random data injection attack is slightly faster than that under constant data injection attack. The underlying reason is that: with a higher probability the random data injection will provide more helpful clock information to accelerate the convergence. The performance of SATS under different attack frequencies is plotted in Fig. 6(b), where the attack strategy is random data injection. It is observed that SATS can defend different attack strategies as well as different attack frequencies.

D. System Insights

In order to further investigate how SATS utilizes the messages from attack nodes, we consider the random graph with $n = 50$ and $m = 9$, where the attack nodes conduct random data injection attack. We compare the performance of SATS for two scenarios. For the first scenario, we normally conduct SATS over the network without advanced knowledge about the attack nodes. However, for the second scenario, we assume all attack nodes have been identified, and we ignore any message from those attack nodes. From Fig. 7(a), it can be observed that the converging speed with the messages from attack nodes is indeed faster than that without utilizing any message from attack nodes. We also plot the accumulative number of accepted messages that are omitted from attack nodes in Fig. 7(b). It can be seen that SATS does utilize the corrupted messages from the attack nodes at the first stage to accelerate the converging speed. And once the time synchronization has been achieved, almost no more corrupted message will be accepted by the safe nodes.

Then, we further investigate how the communication delay affects the synchronization accuracy of SATS, and we also consider the random graph with $n = 50$ and $m = 9$, where the attack nodes conduct random data injection attack. We assume that both the hardware and logical clock checking processes have been revised as described in Section V-F, and the initial clock parameters setting are the same as the above simulations. The performance of SATS under communication delay with different upper bounds (parameter d in (25)) is given in Fig. 8. It is observed that under communication delay with different upper bounds setting, SATS can still converge with exponentially fast speed, and the convergence speed is affected slightly by the delay upper bounds. However, as pointed out in Section V-F, the synchronization cannot be achieved completely by SATS when there are communication delays. As shown in Fig. 8(b), the synchronization accuracy will increase with the upper bound

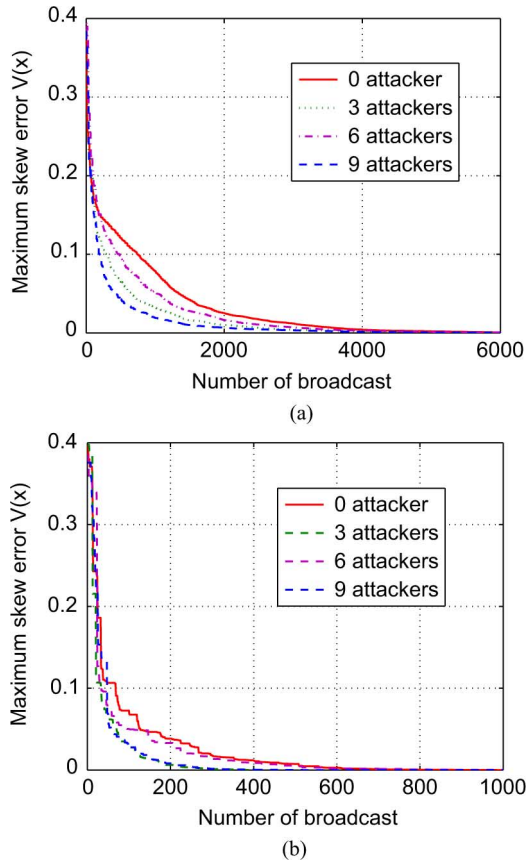

Fig. 5. The performance of SATS with different number of attacks (a) In ring network (b) In random network.

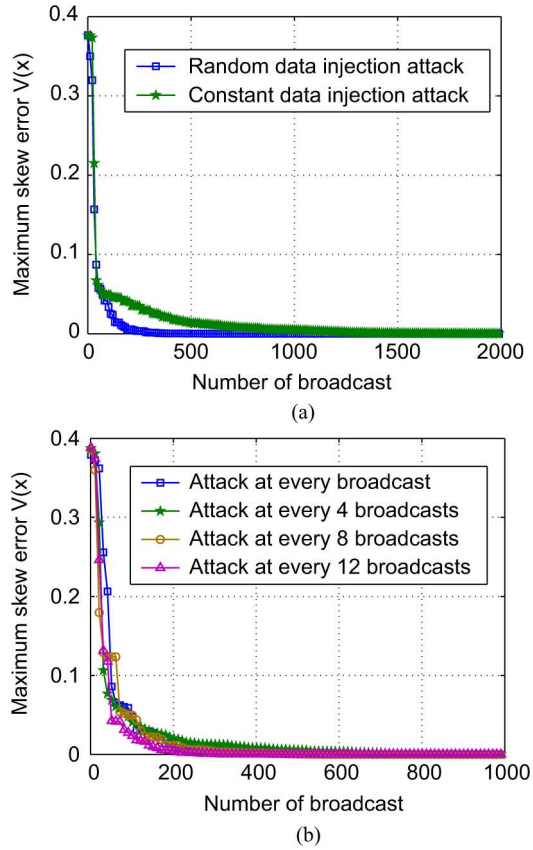

Fig. 6. The performance of SATS (a) Different attack strategies (b) Different attack frequencies.

of communication delay d, e.g., the maximum skew error $V(x)$ is about 1.8×10^{-3} when $d=10^{-3}$, and is about 2×10^{-4} when $d=10^{-4}$. Thus, SATS is robust against the communication delay, and the synchronization error can achieve the same order as the upper bound of communication delay. Additionally, there exist some communication delay compensation techniques proposed in, e.g., [1], [3], [33], [36], [37], which can be utilized to handle the communication delay for SATS, such that SATS obtains a better performance.

VII. Conclusions

In this paper, we have studied how to defend the average consensus-based time synchronization protocol in wireless sensor networks under message manipulation attacks. Through both theoretical analysis and simulations, we show that traditional ATS is highly vulnerable to the message manipulation attacks. Then, we propose the hardware clock checking process for avoiding the hardware clock reading corruption of attackers, and the logical checking process for dynamically constraining the attacker's logical clock corruption. We incorporate such checking processes into the traditional ATS to establish the SATS protocol, under which, both the skew and offset compensation converge exponentially. Our analysis and simulation results show that with SATS the attackers may unconsciously help to increase the time synchronization speed of the network.

Appendix A
The Proof of Lemma 1

Proof: Note that the information sets $\Theta_{i_0}(t_{i_0})$ and $\Theta_{i_1}(t_{i_1})$ are created and authenticated respectively by nodes i_0 and i_1, which can not be modified by the node i. Hence, condition c_1 ensures that nodes i, i_0 and i_1 are different from each other. Node j will indeed think that the hardware clock reading $\tau_i(t_\nu^i)$ in the information set $\Theta_\nu(t_\nu)$ has passed the hardware clock checking process of node ν for $\nu=i_0,i_1$, thus it will use $\tau_i(t_\nu^i)$ as a linear function of t_ν^i, i.e., $\tau_i(t_\nu^i)=a_it_\nu^i+b_i$. Hence, from $\frac{\tau_i(t_i^\nu)-\tau_i(t_\nu^i)}{\tau_\nu(t_\nu)-\tau_\nu(t_\nu^i)}=a_{\nu i}$, where $\tau_i\left(t_\nu^i\right.$, $\tau_\nu(t_\nu)$ and $\tau_\nu(t_\nu^i)$ are in $\Theta_\nu(t_\nu)$, we have $\tau_i(t_i^\nu)=a_it_i^\nu+b_i$. It follows from the definitions of t_i^ν that $t_i^\nu=t_\nu,\nu=i_0,i_1$. Since $\tau_i(t_i)-\tau_i(t_i^\nu)\le\tilde{T}$, $\tau_i(t_\nu)\in[\tau_i(t_i)-\tilde{T},\tau_i(t_i)]$. Meanwhile, τ_i should pass the hardware clock checking process, which leads to that τ_i is a linear function, thus (20) holds from condition c_2. Hence, c_1 and c_2 ensure that $\Theta_{i_0}(t_{i_0})$ and $\Theta_{i_1}(t_{i_1})$ are received by node i from two different neighbor nodes i_0 and i_1 in time interval $[t_i-\frac{\tilde{T}}{(1-\varrho)},t_i]$. ■

Appendix B

The Proof of Theorem 2: Two Lyapunov functions are provided to prove the convergence of SATS. The first one is a general Lyapunov function which is given by

$$V(z)=\max(z)-\min(z), \tag{26}$$

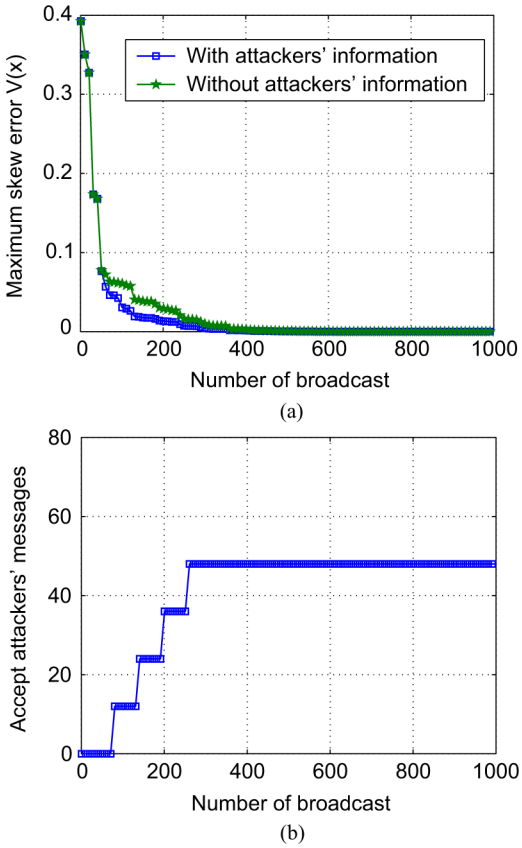

Fig. 7. The insights of SATS (a) Performance of SATS (b) Credible attackers' messages.

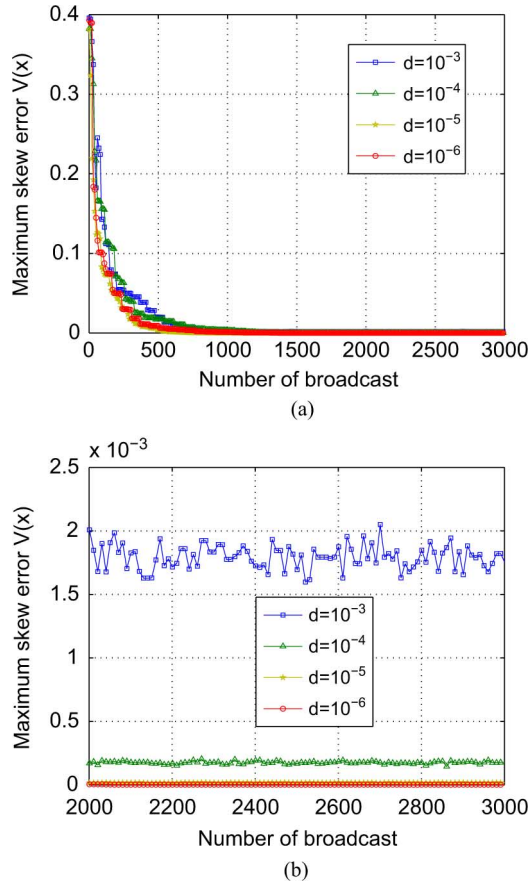

Fig. 8. The performance of SATS under different communication delay bounds (a) 0 to 3000 (b) 2000 to 3000.

where $z \in \mathcal{R}^n$, $\max(z) = \max_{i=1}^n\{z_i\}$ and $\min(z) = \min_{i=1}^n\{z_i\}$. The second one is a new Lyapunov function, which is defined as

$$\begin{aligned}\tilde{V}(x(t)) &= \max_{t'\in[t-\Delta t,t]} x(t') - \min_{t'\in[t-\Delta t,t]} x(t') \\ &= x^t_{max} - x^t_{min},\end{aligned} \tag{27}$$

where $\Delta t = \frac{\tilde{T}}{1-\varrho}$ when $t \geq \frac{\tilde{T}}{1-\varrho}$ and otherwise $\Delta t = 0$. Intuitively, $\tilde{V}(x(t))$ denotes the maximum difference among all safe nodes' logical clock skews within time interval $[t-\Delta t, t]$.

Consider that the logical clock skew vector $x(t)$ is updated by SATS. Two important lemmas of $V(x(t))$ and $\tilde{V}(x(t))$ are given as follows. The following lemma is given to guarantee that $\tilde{V}(x(t))$ is a decreasing function and $V(x(t))$ is bounded by $\tilde{V}(x(t))$.

Lemma B.1: By SATS, $\tilde{V}(x(t))$ is a non-increasing function and

$$V(x(t+t_0)) \leq \tilde{V}(x(t)), t_0 \geq 0. \tag{28}$$

Proof: Let node i be an attack node. In SATS, a safe node j will use the information received from node i to update its clock only when the following inequality holds $x_{i_0}(t_{i_0}) \leq x_i(t_i) \leq x_{i_1}(t_{i_1})$. Meanwhile, there are not any two attack nodes neighboring with each other, thus i_0 and i_1 should be the safe nodes. It follows that $x^{t_i}_{min} \leq x_i(t_i) \leq x^{t_i}_{max}$. Note that the clock update of each node in SATS is an averaging process when the node receives information from neighbor nodes. Thus, we have

$$x^{t_i}_{min} \leq x_j(t_i^+) \leq x^{t_i}_{max}, \tag{29}$$

for the safe node j after each update at time t_i. Clearly, (29) also holds when node i is a safe node. Hence, (29) holds for any safe node j after each update, which means that $\tilde{V}(x(t))$ is a non-increasing function and (28) holds. ■

In the next lemma, it is guaranteed that $\tilde{V}(x)$ a strictly decreasing function within a certain time interval.

Lemma B.2: By SATS, there exists $\epsilon > 0$ such that

$$\tilde{V}(x(t+t_1)) \leq (1-\epsilon)\tilde{V}(x(t)), t_1 \geq \frac{n\tilde{T}}{1-\varrho}. \tag{30}$$

Let $g(z_1, z_2, \rho)$ be a function of variables z_1, z_2 and ρ, which satisfies $g(z_1, z_2, \rho) = \rho z_1 + (1-\rho)z_2$, where $\rho \in [\rho_v^{k_1}(1-\rho_v)^{k_2}, 1]$, and where $0 \leq k_1, k_2 \leq M$ and M is a constant which is the most updating times among safe nodes in time interval $[t, t_1]$. This function is provided to depict the average updating process of SATS, and used as an auxiliary function which helps to prove the monotonicity of the function $\tilde{V}(x(t))$.

Divide time interval $[t, t+t_1]$ into $[t, t+\frac{\tilde{T}}{1-\varrho}]$, $[t+\frac{\tilde{T}}{1-\varrho}, t+\frac{2\tilde{T}}{1-\varrho}]$, $\ldots$, $[t+\frac{(n-1)\tilde{T}}{1-\varrho}, t+t_1]$. We first prove that after nodes i and j have comunicated with each other, the values of both x_i and x_j can be expressed as the function g defined above. For

$t_i, t_j \in [t, t+\frac{\tilde{T}}{1-\varrho}]$, assume that safe node j has received message from its safe neighbor node i at time t_i before its broadcasting at time t_j. It follows that

$$x_j(t_i^+) = \rho_v x_j(t_i) + (1-\rho_v)x_i(t_i) \tag{31}$$

and

$$x_{j_0}(t_{j_0}) \le x_i(t_i) \le x_{j_1}(t_{j_1}). \tag{32}$$

Note that node j receives information from any other neighbor node l at time t_l for $t_l \in [t_i, t_j]$, it has

$$x_j(t_l^+) = \rho_v x_j(t_l) + (1-\rho_v)x_l(t_l). \tag{33}$$

Since $x_l(t_l)$ in (33) satisfies $x_{min}^t \le x_l(t_l) \le x_{max}^t$, combining (31) and (33), it yields that there exists $x_s(t') \in [x_{min}^t, x_{max}^t]$ such that

$$x_j(t') = \rho_1 x_i(t_i) + (1-\rho_1)x_s(t') = g(x_i(t_i), x_s(t'), \rho_1), \tag{34}$$

for $t' \in [t_i, t_j]$, where $\rho_1 = \rho_v^k(1-\rho_v)$ and k is the times of update of node j in time interval $[t_i, t_j]$. From (32) and (35), after broadcast of node j at time t_j, we have

$$x_j(t_j^+) = \rho_2 x_i(t_i) + (1-\rho_2)x_s(t_j) = g(x_i(t_i), x_s(t_j), \rho_2),$$

where $\rho_v^k(1-\rho_v) \le \rho_2 \le 1$ and $x_s(t_j) \in [x_{min}^t, x_{max}^t]$. Meanwhile, node i will update its clock based on the information received from node j, and then

$$x_i(t_j^+) = \rho_v x_i(t_j) + (1-\rho_v)x_j(t_j^+) = g(x_i(t_i), x_s(t_j^+), \rho_3),$$

where $\rho_3 = \rho_v^k(1-\rho_v)^2$ and $x_s(t_j^+) \in [x_{min}^t, x_{max}^t]$, and

$$x_{i_0}(t_{i_0}) \le g(x_i(t_i), x_s(t_j), \rho_2) \le x_{i_1}(t_{i_1}).$$

Then, we prove that the logical clock skews of all safe nodes can be expressed as the function g at time t' for $t' > t_1$. As the communication is reliable, nodes i and j can communicate with each other successfully in any time $[t', t'+\frac{\tilde{T}}{1-\varrho}]$, where $t' \ge t_j$. It follows that

$$x_l(t') = g(x_i(t_i), x_s, \rho_a), \tag{35}$$

and

$$x_{l_0}(t_{l_0}) \le g(x_i(t_i), x_s, \rho_b) \le x_{l_1}(t_{l_1}), \tag{36}$$

for $t_1 \ge t' \ge t_j$ and $l = i, j$ (where x_s, ρ_a and ρ_b in g may be different for different nodes but should satisfy $x_s \in [x_{min}^t, x_{max}^t]$ and $\rho_a, \rho_b \in [\rho_v^{k_1}(1-\rho_v)^{k_2}, 1]$). It means that for both nodes i and j, we always have (35) and (36) for $\forall t' \in [t_j, t_1]$. Thus, after time $t+\frac{\tilde{T}}{1-\varrho}$, there are at least two safe nodes with (35) and (36). Note that every node can receive information successfully from safe neighbor nodes in any time interval $[t', t'+\frac{\tilde{T}}{1-\varrho}]$, and the updating of each safe node is an averaging process. Hence, for i or j's safe neighbor node k, if it updates based on received message from node i or j at time t_k for $t_k > t_j$, then it has (35) and (36) for $l = k$ when $t' \le t_1$. Note that node k can receive information successfully from the node i or j in time interval $[t+\frac{\tilde{T}}{1-\varrho}, t+\frac{2\tilde{T}}{1-\varrho}]$, which means that there are at least three safe nodes with (35) and (36) after time $t+\frac{2\tilde{T}}{1-\varrho}$. Similarly, it can follow that there are at least n safe nodes with (35) and (36) after time $t+\frac{(n-1)\tilde{T}}{1-\varrho}$, namely, (35) and (36) hold for all safe nodes.

Therefore, for $\forall t', t'' \in [t+\frac{(n-1)\tilde{T}}{1-\varrho}, t+t_1]$ and $j_1, j_2 \in \mathcal{V}_s$,

$$\begin{aligned}
&|x_{j_1}(t') - x_{j_2}(t'')| \\
&= |g(x_i(t_i), x_s(t'), \rho') - g(x_i(t_i), x_s(t''), \rho'')| \\
&\le (1-\rho)(x_{max} - x_{min}) \\
&\le (1-\rho_v^M(1-\rho_v)^M)(x_{max}^t - x_{min}^t),
\end{aligned}$$

where $\rho = \min\{\rho', \rho''\}$. It follows that

$$\tilde{V}(x(t+t_1)) \le (1-\rho_v^M(1-\rho_v)^M)\tilde{V}(x(t)),$$

which completes the proof. ■

Combining Lemma B.1 with Lemma B.2, it yields that $x(t)$ will exponentially converge to 0, thus Theorem 2 is proved.

REFERENCES

[1] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Netw.*, pp. 281–323, 2005.

[2] Q. Li and D. Rus, "Global clock synchronization in sensor networks," presented at the Infocom, 2004.

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," presented at the SenSys, 2003.

[4] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," presented at the SenSys, 2004.

[5] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 214–225, Feb. 2006.

[6] A. Giridhar and P. R. Kumar, "Distributed clock synchronization over wireless networks: Algorithms and analysis," presented at the CDC, 2006.

[7] J. He, P. Cheng, L. Shi, and J. Chen, "Clock synchronization for random mobile sensor networks," presented at the CDC, 2012.

[8] S. Philipp and W. Roger, "Gradient clock synchronization in wireless sensor networks," presented at the IPSN, 2009.

[9] L. Schenato and F. Fiorentin, "Average timesynch: a consensus-based protocol for time synchronization in wireless sensor networks," *Automatica*, pp. 1878–1886, 2011.

[10] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in wsns: A maximum-value-based consensus approach," *IEEE Trans. Autom. Control*, 2013, DOI:10.1109/TAC.2013.2286893, accepted for publication.

[11] J. He, P. Cheng, L. Shi, and J. Chen, "Time synchronization in wsns: A maximum value based consensus approach," presented at the CDC, 2011.

[12] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "Optimal synchronization for networks of noisy double integrators," *IEEE Trans. Autom. Control*, vol. 56, no. 5, pp. 1146–1152, May 2011.

[13] B. Choi, H. Liang, and X. Shen, "Dcs: Distributed asynchronous clock synchronization in delay tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 491–504, 2012.

[14] R. Carli, E. Elia, and S. Zampieri, "A pi controller based on asymmetric gossip communications for clocks synchronization in wireless sensors networks," presented at the CDC, 2011.

[15] X. Cao, J. Chen, Y. Zhang, and Y. Sun, "Development of an integrated wireless sensor network micro-environmental monitoring system," *ISA Trans.*, pp. 247–255, 2008.

[16] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *IEEE Trans. Signal Processing Mag.*, vol. 28, no. 1, pp. 124–138, Jan. 2011.

[17] S. Ganeriwal, S. Capkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc. WiSe*, 2005.
[18] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commun. ACM*, pp. 53–57, 2004.
[19] S. Zhu, S. Setia, and S. Jajodia, "Leap+: efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. on Sensor Networks*, pp. 500–528, 2006.
[20] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 2, pp. 395–408, Feb. 2006.
[21] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "Tinysersync: Secure and resilient time synchronization in wireless sensor networks," presented at the CCS, 2006.
[22] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," presented at the MASS, 2005.
[23] S. Ganeriwal, C. Popper, S. Capkun, and M. B. Srivastava, "Secure time synchronization in sensor networks," *ACM Trans. Inf. Syst. Security*, 2008.
[24] X. Hu, T. Park, and K. G. Shin, "Attack-tolerant time-synchronization in wireless sensor networks," presented at the Infocom, 2008.
[25] 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," presented at the Infocom, 2009.
[26] R. Robles, J. Haas, J. Chiang, Y.-C. Hu, and P. R. Kumar, "Secure network-wide clock synchronization in wireless sensor networks," presented at the CDC, 2010.
[27] D. Huang, K. You, and W. Teng, "Secured flooding time synchronization protocol," presented at the MASS, 2011.
[28] M. Rahman and K. Khatib, "Secure time synchronization for wireless sensor networks based on bilinear pairing functions," *IEEE Trans. Parallel Distrib. Syst.*, 2011, to be published.
[29] C. Benzaid, A. Saiah, and N. Badache, "Secure pairwise broadcast time synchronization in wireless sensor networks," presented at the DCOSS, 2011.
[30] R. Wang, W. Du, X. Liu, and P. Ning, "Shortpk: A shortterm public key scheme for broadcast autentication in sensor networks," *ACM Trans. Sensor Netwo.*, pp. 19–29, 2009.
[31] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Secure and efficient time synchronization in heterogeneous sensor networks," *IEEE Trans. Veh. Technol.*, pp. 2387–2394, 2008.
[32] *TmoteSky. Tmote sky data sheet*. New York: Moteiv Corporation, 2004.
[33] J. Kim, J. Lee, E. Serpedin, and K. Qaraqe, "Robust clock synchronization in wireless sensor networks through noise density estimation," *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3035–3047, Jul. 2011.
[34] L. Mei and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5404–5414, Nov. 2011.
[35] Q. M. Chaudhari, E. Serpedin, and K. Qaraqe, "On maximum likelihood estimation of clock offset and skew in networks with exponential delays," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1685–1696, Apr. 2008.
[36] L. Mei and Y.-C. Wu, "Low-complexity maximum-likelihood estimator for clock synchronization of wireless sensor nodes under exponential delays," *IEEE Trans. Signal Process.*, vol. 59, no. 10, pp. 4860–4870, Oct. 2011.
[37] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2010.
[38] Q. Yan, M. Li, W. Lou, and Y. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," presented at the Infocom, 2012.
[39] 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 Distrib. Syst.*, vol. 23, no. 1, pp. 32–43, Jan. 2012.

Jianping He is currently a Ph.D. candidate 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.

Peng Cheng (M'10) received the B.E. degree in Automation, and the Ph.D. degree in Control Science and Engineering in 2004 and 2009 respectively, both from Zhejiang University, Hangzhou, P.R. China. Currently he is Associate Professor with Department of Control Science and Engineering, Zhejiang University. He serves as the publicity co-chair for IEEE MASS 2013. His research interests include networked sensing and control, cyber-physical systems, and robust control.

Ling Shi received his B.S. degree in Electrical and Electronic Engineering from the Hong Kong University of Science and Technology in 2002 and Ph.D. degree in Control and Dynamical Systems from California Institute of Technology in 2008. He is currently an Assistant Professor at the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology. His research interests include networked control systems, wireless sensor networks and distributed control.

Jiming Chen (M'08–SM'11) received B.Sc. degree and Ph.D. degree both in Control Science and Engineering from Zhejiang University 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. Currently, he is a full professor with 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, China. 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 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,'11,'13, IEEE SECON'11,'12 IEEE INFOCOM'11,'12,'13, etc.