This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

A Secure Time Synchronization Protocol against Fake Timestamps for Large Scale Internet of Things

Tie Qiu, *Senior Member, IEEE,* Xize Liu, Min Han, *Senior Member, IEEE,* Huansheng Ning, *Senior Member, IEEE and* Dapeng Oliver Wu, *Fellow, IEEE*

***Abstract*—For large scale internet of things (IoT) which located in the hostile environment where exists malicious nodes, the security of time synchronization is a critical and challenging issue. The malicious sensor nodes could decrease the accuracy of the whole network by broadcasting fake timestamp messages. In this paper, we propose a secure time synchronization model for large scale IoT. In this model, a node utilizes its father node and grandfather node to detect the malicious node. By employing the model, a spanning tree topology which synchronizes to the reference nodes can be constructed hop by hop. Then a Secure Time Synchronization Protocol (STSP) is developed to against fake timestamps, which adopts the secure model. We use NS2 as the simulation tool to evaluate our protocol, and compare the impact of fake timestamps in various circumstances with the pervious protocols TPSN and STETS. The experiment results show that our protocol is effective to prevent attacks from malicious nodes.**

***Index Terms*—Security, time synchronization, fake timestamps, Internet of Things.**

I. Introduction

THE Internet of Things (IoT) allows the objects collect and data exchange. It consists of many novel networking techniques. [1], [2], [3], [4]. With the assist of sensors and actuators, IoT is widely applied in industrial automation [5], [6], mobile object tracking [7], [8], environmental monitoring [9], etc. Time synchronization is a critical issue for operations such as scheduling sleeping [10], power mangement and speed calculation in IoT. Because these operations need the distributed sensor nodes to establish a global time to accomplish the tasks collaboratively. Many protocols are designed to improve accuracy [11], [12] and energy efficiency [13], [14] of time synchronization. However secure timestamp is the base of precise synchronization. Therefore security problem [15], [16], [17] becomes increasingly important in IoT.

A wide variety of IoT systems such as smart grids, intelligent transportation, smart cities and virtual power plants need a global time to operate normally. It is necessary to establish a secure and reliable time synchronization system for these large scale IoT systems. These IoT systems need the distributed units to collaboratively complete information collection, remote monitoring, automatic management, etc. The security of time synchronization is significant to these distributed IoT systems. Each distributed node maintains its own local clock, therefore the time of all nodes is inconsistent. Time synchronization is vital to scheduling mechanism, for example the TDMA protocol. After the time synchronization, the fixed communication time is allocated to each node according to the slot. If the accuracy of time synchronization is too low, multiple nodes will send messages at the same time, which results in the network conflict. Furthermore, the design of positioning and target tracking program also needs the time synchronization. For the mobile target, timestamps need to be marked in some important messages. Time synchronization is essential in energy management mechanism as well. In order to control energy consumption, the nodes are put to sleep when they do not work. For distributed systems, nodes cannot sleep or awake at the same time without time synchronization.

In the hostile environment where malicious nodes are deployed for attack, sensor nodes cannot identify whether the received messages are fake or not. Some nodes might utilize the wrong timestamps to synchronize the time. Even worse, the normal sensor node would broadcast wrong timestamps after synchronized to the malicious nodes. As a result, few malicious nodes could affect the time synchronization of the whole network. Therefore how to design a secure mechanism for the IoT to against fake timestamps is significant. In recent years many researchers focus on relevant research [18], [19], [20] to improve the security of time synchronization as well as maintain the performance in hostile environment.

Receiver-to-Receiver Protocol (RRP) [21], [22], [23] and Sender-to-Receiver Protocol (SRP) [24], [25], [26], [27] are two classical models which are widely used in time synchronization. SRP model aims to achieve high accuracy time synchronization between two neighboring nodes. A sensor node synchronizes to the reference node according to the estimated clock offset which calculated by four timestamps. However, the time synchronization mechanisms which directly use SRP models are vulnerable to malicious nodes. Because sensor nodes may use the fake timestamps sent by malicious nodes to compute the clock offset. Especially, the attacks to network time synchronization can disturb the execution of tasks, make the message disordered, which decreases the entire performance of the whole networks. In this paper, a secure time synchronization model is developed to detect the fake timestamps. And then STSP is proposed based on the

T. Qiu, X. Liu are with the School of Software, Dalian University of Technology, Dalian, Liaoning, China, 116620 (e-mail: qiutie@ieee.org; liuxize@mail.dlut.edu.cn).

M. Han is with Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning, China, 116023 (e-mail: minhan@dlut.edu.cn).

H. Ning is with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China, 100083 (ninghuansheng@ustb.edu.cn).

D. O. Wu is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, USA, 32611 (e-mail: wu@ece.ufl.edu).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

model and our previously work STETS [28] for secure time synchronization. The contributions of this paper are as follows:

- A secure time synchronization model is given. The model utilizes three nodes to synchronize the clock, including the node to be synchronized, its father node and grandparent node. A sensor node calculates the clock offset between its father node and grandfather node. And then it determines whether the timestamps broadcasted from father node is true according to the clock offset.
- We propose STSP, a secure time synchronization protocol. It combines our secure model and RRP model to against fake timestamps and reduces the impact of malicious nodes.
- We use NS2 to simulate STSP and evaluate the performance in various circumstances. The simulation results show that our protocol performs better security compared with TPSN [25] and STETS [28]. It prevents more nodes from synchronizing to the malicious nodes.

The rest of this paper is organized as follows: The related work and problem statement are given in Section 2. In Section 3, we present the assumption of STSP, and introduce the design of secure time synchronization model in detail. In Section 4, the algorithms are designed for secure time synchronization. In Section 5, we do the simulation with NS-2 and evaluate the performance of STSP compared with TPSN and STETS. The security performance is evaluated from synchronization accuracy and the percentage of nodes which synchronized to the reference clock. Finally, we offer our conclusion of the paper and give the future work.

II. Related work and problem statement

A. Related Work

Many approaches have been studied for time synchronization. IEEE 1588 Precision Clock Synchronization Protocol (PTP) is designed for local systems which require accuracies beyond those attainable using NTP. It is widely applied to IoT systems like smart grid. It describes a hierarchical master-slave architecture for clock distribution. The process of time synchronization of PTP is similar to SRP model. It utilizes four timestamps to calculate the time offset between master clock and slave clock. Then the slave clock utilizes the offset to compensate the local clock and synchronizes to the master clock. However, when the slave clock needs multiple hops to synchronize to the master clock, PTP cannot resist the impact of malicious timestamps. Therefore the accuracy of time synchronization will be greatly reduced. Compared with PTP, our proposed STSP is effective to prevent attacks of fake timestamps from malicious nodes in multi-hop networks.

A kind of method consists of centralized protocols [25], [29], [30], which relys on certain reference nodes. All nodes synchronize to reference nodes by constructing a tree-based topology. TPSN [25] achieves network-wide time synchronization by employing SRP model. The process can be divided into two parts. Above all, a spanning tree structure is built in sensor network, then sensor nodes exchange timestamps message along each edge. As a result, all nodes can be synchronized to the reference nodes. FCSA [29] achieves network-wide time synchronization by selecting reference nodes in the network. FCSA improves the synchronization accuracy and scalability with slow-flooding. It focuses on reducing the time error caused by clock drift and decreasing the time that required to achieve network-wide time synchronization.

Some other technical methods can be classified as distributed time synchronization [31], [32], [33], [34]. All nodes collect the time information of neighboring nodes within communication range to compensate the clock offset. This approach outperforms centralized protocols in terms of robustness.

However, these proposed protocols assume that sensor networks located in a benign environment. Security problem has been recognized as a challenging and significant issue, many protocols provide high security to against various attacks. For example in TinySeRSync [35], the authors propose a single-hop security time synchronization protocol, which utilizes the hardware to authenticate MAC layer timestamps. And then it employs the μTESLA protocol for secure message broadcasting. TinySeRSync ensures the sensor networks against external attacks and resiliently against compromised nodes. Two solutions for secure time synchronization are proposed in [36] to against delay attacks. First of all, GESD algorithm was designed to detect multiple outliers produced from malicious nodes. The second method utilizes a threshold derived algorithm to filter out the outliers. After collecting the time deviations among a series of nodes, these two methods are employed to distinguish outliers from these deviations and accommodate the delay attacks. In [37], authors propose a security mechanism to address the security vulnerability for FTSP. It considers the attacks to root node and common nodes in the topology. A reference node selecting mechanism and data filter algorithms are proposed to defend against the attacks from malicious nodes. The attacks include the modifying of global time, seq number and the frequency of sending packets. SATS [38] is proposed to defend against message manipulation attacks on average-consensus-based time synchronization protocol ATS. By utilizing a hardware clock checking process, the message manipulation for parameters can be defended. Furthermore, logical checking process is adopted to constrain the impact of attackers. He et al. propose a distributed and secure time synchronization protocol SMTS [18]. It utilizes the maximum consensus-based approach to invalidate message manipulation attacks, which concerns with clock skew and offset. The hardware clock and logical clock checking processes are used to detect malicious nodes, and then the fake timestamps are ignored. ATSP [39] can deal with the attack of packet faking and delay. It mainly exploiting temporal correlations among nodes to achieve high attack-tolerance. In [40], authors propose an approach to guarantee secure pairwise and group time synchronization. By checking the end-to-end delays exceed a certain threshold, it can defense DoS attacks.

In this paper, we focus on designing a tree-based protocol with secure scheme for clock synchronization to against fake timestamps.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

B. Problem Statement

Time synchronization plays an important role in the applications of IoT. Several efficient time synchronization protocols with high accuracy have been proposed. Most of them do not consider the security so that they cannot satisfy the applications in hostile environments. For tree-based time synchronization model, the root node can be connected to device like GPS. Furthermore, considering the computing ability of sensor nodes is limited in IoT. The time complexity of synchronizing all nodes' time in tree-based protocols is further lower than that of distributed protocols. Because distributed protocols need multi-iteration to calculate a precise global time, while tree-based networks only need several messages exchanging. However, there are no secure tree-based protocols to against fake timestamps according to our investigation. Thus a secure tree-based clock synchronization strategy is expected to ensure the precision of synchronization.

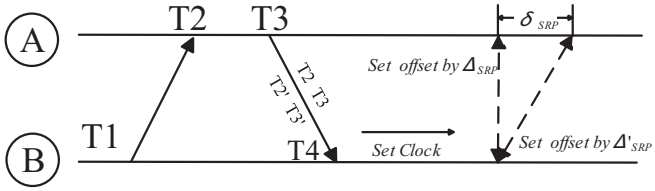


Fig. 1: SRP model.

SRP model and RRP model are classical in tree-based time synchronization protocols. The detail process of SRP model is shown in Fig. 1. Above all, the neighboring node (node B) broadcasts a *Sync* message and records the timestamp $T1$ meanwhile. When the reference node (node A) receives the *Sync* message, timestamp $T2$ is recorded. At time $T3$, node A broadcasts an *Ack* message which carries timestamps $T2$ as well as $T3$. If the node B receives the *Ack* message, it records timestamp $T4$. The clock offset can be calculated by these four timestamps according to Eq. 1.

$$\Delta_{SRP} = \frac{(T2-T1)-(T4-T3)}{2} \tag{1}$$

$$\Delta'_{SRP} = \frac{(T2'-T1)-(T4-T3')}{2} \tag{2}$$

$$\delta_{SRP} = \Delta_{SRP} - \Delta'_{SRP} = \frac{(T2-T2')+(T3-T3')}{2} \tag{3}$$

We suppose that the node A is malicious and broadcasts fake timestamps $(T2', T3')$. Then node B synchronizes with node A with the false offset Δ'_{SRP} given in Eq. 2. According to Eq. 3, the difference value δ_{SRP} of node B 's time error in normal and malicious conditions can be calculated. For the multi-hop pairwise clock synchronization protocols that establish a spanning tree structure [25], [26], [27], all nodes synchronize the clock along these paths by utilizing SRP model. A single malicious node can produce a series of errors. In addition, the location of malicious node is relative to the rate of false time synchronization. The closer malicious node located to root node, the more normal nodes will be impacted.

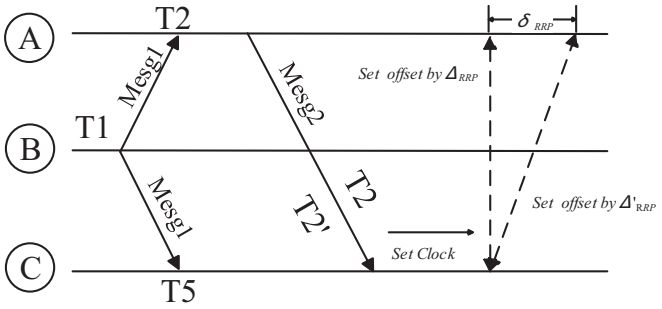


Fig. 2: RRP model.

The process of RRP model is shown in Fig. 2. Node C can synchronize to node A without broadcasting any message. Above all, node B broadcasts a $Mesg1$ message at $T1$. Then both node A and node C record the timestamps $T2$ and $T5$ respectively when receives the $Mesg1$. After that node A sends a $Mesg2$ message which carries $T2$ to node A. Node A can calculate the offset with node C according to Eq. 4. Considering the condition that node A is malicious, node B receives the fake timestamp $T2'$ and calculates the false offset Δ'_{RRP} given in Eq. 5. The difference value δ_{RRP} of node B is calculated by Eq. 6. RBS [21] is a time synchronization protocol based on RRP model. The nodes receive timestamp messages from reference nodes to synchronize to each other. This method limits the impact of malicious nodes because the affected nodes do not broadcast any fake timestamps. What's more, the energy consumption can be decreased greatly by using the RRP model.

$$\Delta_{RRP} = T2 - T5 \tag{4}$$

$$\Delta'_{RRP} = T2' - T5 \tag{5}$$

$$\delta_{RRP} = \Delta_{RRP} - \Delta'_{RRP} = T2 - T2' \tag{6}$$

The protocols which utilize SRP model maybe introduce large-scale synchronization errors. For the protocols based on RRP model, the synchronization errors are limited but cannot be corrected. To improve security for tree-based protocols that utilize these two models, a secure synchronization model is proposed to against fake timestamps. Then we develop STSP for secure time synchronization.

III. Model

A. Assumption

We assume that each sensor nodes has a fixed identify (ID). A secure and stable node is set as the clock source, which broadcasts the correct timestamps. Sensor nodes consist of normal nodes (NNs) and malicious nodes (MNs). MNs broadcast fake timestamps to the network. Furthermore, We divide the sensor nodes into undefined nodes (UNs), backbone nodes (BNs) and passive nodes (PNs). Each node has inner timers and maintains a logical time model, which is shown in Eq. 7.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

$$L_i(t) = \alpha * H_i(t) + \beta \tag{7}$$

α is the drift among nodes, β is the initial offset, t is the time of root node. $L_i(t)$ represents the logical time of node i at standard time t. $H_i(t)$ stands for the hardware time of node i at the standard time t. Before the synchronization, α is initialized to 1 and β is set to 0. In this paper, we assume that malicious nodes also maintain the same logical clock model, but the value of the parameters α and β are abnormal so that they broadcast fake timestamps messages. In this paper, father and grandfather are used to express the relationship between nodes, such as in Fig. 3. Node B is node C's father node, and node A is node C's grandfather node. The symbols used in this paper is illustrate in Table I.

TABLE I: Symbols and description.

Symbol	Description
Ti	i-th timestamp
Ti'	i-th fake timestamp
Δ_m	time offset calculated by mode m (m can be *SRP* or *RRP*)
Δ'_m	false time offset calculated with false timestamps by mode m
δ_m	difference between Δ_m and Δ'_m
$\Delta(x,y)$	time offset between node x and node y
λ	the maximum time offset between two synchronized nodes
$L_i(t)$	logical time of node i at standard time t
$H_i(t)$	hardware time of node i at standard time t

B. Main idea

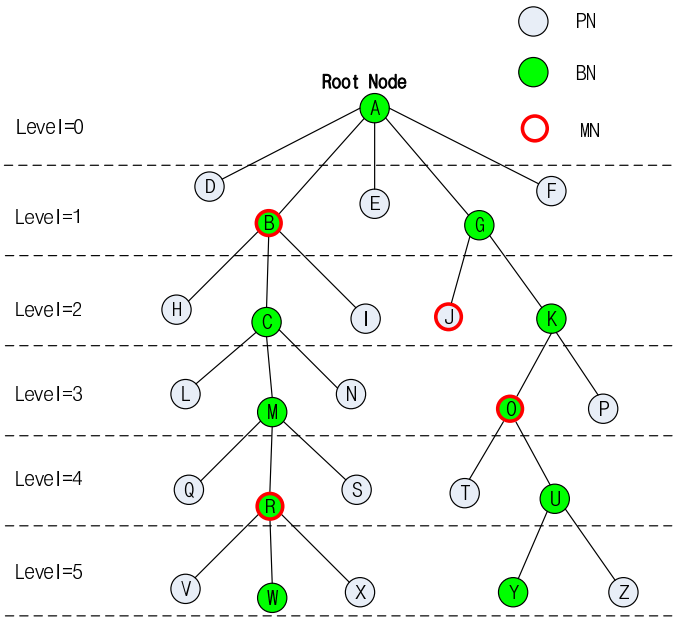


Fig. 3: A spanning tree structure with malicious nodes.

A spanning tree structure is shown in Fig. 3, four MNs are deployed in different levels. These four nodes broadcast fake timestamps. In our pervious work STETS, BNs use SRP model to synchronize the time, and PNs use RRP model to synchronize with BNs. The child nodes of node B all synchronize to false time source due to the attack of fake timestamps. Node O impacts the nodes in level four and five of the right branch.

To decrease the influence of malicious nodes, we introduce a secure time synchronization model depicted in Fig. 4. A node (Node C) utilizes its father node (Node B) and grandfather node (Node A) for time synchronization. In the SRP model, the nodes synchronize to each other directly. Our secure model detects the malicious nodes and avoids synchronizing to the false time. The detail process of our model is as follows, node C takes nodes A and B as reference nodes. Node C broadcasts a *Sync* message at time $T1$. Node B receives the *Sync* message at $T2$ and broadcasts an *Ack* message at $T3$. Both node A and node C can receive the *Ack* message, and record the time as $T4$ and $T6$ respectively. Then node A replies a *Rspnd* message which carries timestamps $T6$ and $T7$. When receives the *Rspnd* message, node B forwards it to node C. Node C records the timestamp $T8$ when it receives the *Fwd* message. Since node C acquires the timestamps $T1$ to $T8$, it calculates the clock offset ($\Delta(C,B)$) between node C and B according to Eq. 8. Similarly, the clock offset ($\Delta(C,A)$) between node A and C can be calculated by Eq. 9.

$$\Delta(C,B) = \frac{(T2-T1)-(T4-T3)}{2} \tag{8}$$

$$\Delta(C,A) = \frac{(T6-T1)-(T8-T7)}{2} \tag{9}$$

What's more, node C can obtain the offset ($\Delta(B,A)$) between father node and grandfather node according to Eq. 10. Local time error presents one hop error. Considering the local time error between synchronized nodes has a threshold value, which can be acquired by multiple tests. The threshold value can be used to detect whether the clock offset between neighboring nodes is outside the normal range. In this model, we utilize the threshold λ to judge whether the father node is malicious or not.

$$\Delta(B,A) = \frac{(T2+T3+T8)-(T4+T6+T7)}{2} \tag{10}$$

When $|\Delta(B,C)| \leqslant \lambda$ is satisfied, it stands for that node B broadcasts the right timestamps. Thus node A can synchronize to node B. If $|\Delta(B,C)| > \lambda$ is satisfied, node B can be treated as a malicious node, then node A synchronizes to the grandfather node C. As a result, this process ensures a node synchronize to the right time source.

Then, STSP is proposed based on the secure model. All nodes are initialized as UNs at first. BNs construct a spanning tree in the topology by employing the secure time synchronization model. Thus a trusted backbone network is constructed hop by hop. PNs only receive messages from BNs and utilize RRP model to synchronize the clock. Therefore, the malicious nodes cannot make a serious impact on the synchronization results. When malicious nodes are detected by our model, BNs make a mark and then inform the child nodes to avoid using the fake timestamps. STSP increases the security of time synchronization by reducing the probability of synchronizing to the malicious nodes.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

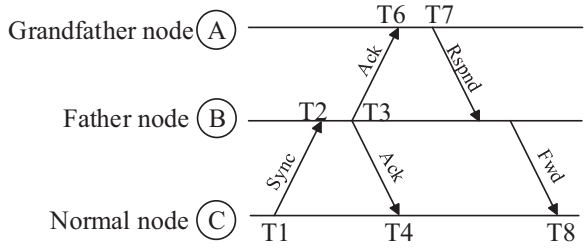


Fig. 4: Time synchronization model.

C. Message format

The message format of STSP is composed by the following seven parts:

DestAddr	SrcID	Type	FID	FTS	STS	L

- DestAddr: destination address of the message. It represents a broadcast message if the value is -1.
- SrcID: the ID of the sender.
- Type: message type. The type of messages including $Init, Sync, Ack, Rspnd$ and Fwd.
- FID: the ID of father node.
- FTS: the first timestamp carried in the message.
- STS: the second timestamp carried in the message.
- L: in *Init*, *Sync* and Fwd messages, it indicates the level of the node in the spanning tree . It is used to store a ID in $Rspnd$. In *Ack* message, it is used to mark whether the sender's father node is a malicious node.

D. Solutions for packet loss

The packet loss and congestion of networks can seriously impact the performance of the proposed secure time synchronization model. Therefore a solution is needed to solve these problems. By utilizing the data packet retransmission mechanism, the impact of packet loss can be avoided. We take Fig. 4 as an example to illustrate the packet retransmission mechanism. The timer is adopted to realize this process. To start the time synchronization, father node B broadcasts an *Init* message to its child node C, meanwhile it sets up a timer $Timer_a$. The initial value is set to 2θ. The situations of packet loss that impact the secure model can be divided into the following types:

- Father node B cannot receive the $Rspnd$ message. In order to avoid this situation, node B sets up the timer $Timer_b$ when broadcasting the *Ack* message. The initial value of the timer is set to 2θ. θ stands for the longest time that requires to send and receive a message between any two neighbouring nodes. If node B doesn't receive the $Rspnd$ message before $Timer_b$ expires. Node B re-sends the Ack message, meanwhile timestamps T3 to T8 are re-recorded. When node C calculates the time offset $\Delta(B,A)$, the time of $Timer_b$ needs to be considered. $\Delta(B,A)$ can be calculated by Eq. 11.

$$\Delta(B,A) = \frac{(T2+2\theta+T3+T8)-(T4+T6+T7)}{2} \quad (11)$$

- Normal node C cannot receive the *Ack* message. Node C sets up a timer $Timer_b$ when broadcasting *Sync* message. The initial value of the timer is set to 2θ. If $Timer_b$ expires and node C doesn't receive the *Ack* message. Node C re-sends the *Sync* message.
- Normal node C cannot receive the Fwd message. Node C sets up a timer $Timer_c$ whose initial value is 4θ when broadcasting *Sync* message. If $Timer_c$ expires and node C doesn't receive the Fwd message, node C re-sends the *Sync* message.

In order to reduce the impact of congestion, the data packets are set with diffenert priority flags. Data packets that used in STSP have the highest priority. When the congestion appears in the network, if a node receives a data packet with the highest priority, it selects a packet with lower priority from the buffer queue and drops it. Therefor, the data packets about time synchronization can be put in the buffer queue and the time synchronization process continues.

IV. Algorithm Design

A. Process of STSP

In this section, we provide the details process of STSP, which is divided into 8 steps.

Step 1: First of all, we choose a node as the time source. It can be connected to device like GPS and broadcasts the correct timestamps. All nodes are initialized to UNs. Then the type of root node changes to BN and the level is set to 0.

Step 2: The root node broadcasts an *Init* message. The UNs which receive this message mark the node as father node and change the level to $L+1$ (L indicates the sender's level). Then they start the timer called Sync Timer (ST) with random values to avoid message collision.

Step 3: If an UN's ST expires, it changes its own node type to BN. And then it sends a *Sync* message and records the time $T1$. The destination address of the message is its father node.

Step 4: When a node receives a *Sync* message, it records the timestamp $T2$.

Case 1. If the destination address of the message is itself and the node type is BN, it broadcasts an *Ack* message at time $T3$.

Case 2. FID in *Sync* message is equal to father node's ID. It sets the node type to PN and cancel the timer ST.

Step 5: If a node receives an *Ack* message:

Case 1. The ID is equal to the FID in the *Ack* message, which indicating that the current node is the father node of sender. It records the timestamp $T6$ when receives the *Ack* message. In order to forward the timestamps $T6$ and $T7$, the variable $TempID$ which records the $DestAddr$ of *Ack* message is used to represent destination address of Fwd message. $TempID$ is carried in the field of L in $Rspond$ and Fwd message. Then at time $T7$, it sends a $Rspond$ message which carries the $T6$, $T7$ and $TempID$ in the field FTS, STS and L respectively. The destination address of $Rspond$ message is the source address of the *Ack* message that just received.

Case 2. If the destination address of *Ack* message is itself and the message is sent by its father node. The timestamp is recorded as T4. If the node is in level one of the spanning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

tree, it utilizes Eq. 8 to calculate clock offset with father mode directly and compensate the clock. Then the node sends an *Init* message.

Case 3. If the destination address of the message is not itself, but the node type is *PN* and the *Ack* message is sent by its father node. The timestamp is recorded as $T5$, and then the FTS value in the message is marked as $T2$. Eq. 8 is used to estimate the time deviation and adjust the logical time of the node.

If the message doesn't satisfy the above three conditions, the node ignores the message.

Step 6: When the *Rspond* is received, the timestamps in FTS and STS are acquired, i.e., timestamps $T6$, $T7$. If the destination address of the *Rspond* message is itself, it sends a Fwd message. The destination address of Fwd message is in L field of *Rspond* message. The Fwd message carries timestamps $T6$ and $T7$ in the field FTS and STS.

Step 7: When a Fwd message is received, the timestamp $T8$ is recorded. If the destination address of the message is itself, the node uses Eq. 9 to calculate the time offset $diff$ between its father node and grandfather node.

Case 1. If $diff$ is less than the threshold value λ (the value of maximum local time error), then it calculates the offset with father node according to Eq. 8.

Case 2. If $diff$ is more than λ, it means father node is a malicious node. Therefore it calculates the offset with grandfather node according to Eq. 9 and synchronizes to grandfather node. Then it broadcasts a *Corr* message to neighboring nodes to correct the *PNs* which synchronize to the malicious node.

Step 8: If the received message type is *Corr* and the current node is *PN* and the father node is as same as the FID in the message, the node adjusts the clock with $diff$.

B. Pseudocode

The pseudocode of the STSP is presented in Algorithms 1 to 4. The preparation of time synchronization is shown in the Algorithm 1, the type of nodes is distinguished by variable *nodeType*. If the node is the root node (Line 1), it becomes a *BN* node and then broadcasts an *Init* message (Lines 2-3). Other nodes are initialized to *UN* (Line 5). *myLevel* indicates the level of the node in spanning tree, which is initialized to 0 (Line 7). The boolean variable *isFatherAtk* identifies whether the node's father node is malicious. *isFatherAtk* is initialized to 0 (Line 8).

Algorithm 1 Initialization For Time Synchronization

```
1: if the node is root node then
2:     nodeType ← BN
3:     broadcast < −1, ID, Init, Null, −1, −1, 0 >
4: else
5:     nodeType ← UN
6: end if
7: myLevel ← 0
8: isFatherAtk ← false
```

Algorithm 2 Time Synchronization Process

```
1: Upon ST expires
2: nodeType ← BN
3: T1 ← CurrentTime()
4: broadcast < myPrntID, ID, Sync, myPrntID, −1, −1, myLevel >
5:
6: Upon receiving < DestAddr, SrcID, Type, FID, FTS, STS, Level >
7: if Type = Init and nodeType = UN then
8:     set up timer ST
9:     myLevel ← Level + 1
10:    myPrntID ← SrcID
11: else if Type = Sync then
12:    T2 ← CurrentTime()
13:    if myPrntID = FID and nodeTypr = UN then
14:        nodeType ← PN
15:        cancel ST
16:    else if DestAddr = ID then
17:        T3 ← CurrentTime()
18:        broadcast < SrcID, ID, Ack, myPrntID, T2, T3, myLevel >
19:    end if
20: else if Type = Ack then
21:    tempTime ← CurrentTime()
22:    if ID = FID then
23:        T6 ← tempTime
24:        tempID ← SrcID
25:        T7 ← CurrentTime()
26:        broadcast < SrcID, ID, Rspond, myPrntID, T6, T7, tempID >
27:    else if DestAddr = ID and SrcID = myPrntID then
28:        goto algorithm 3
29:    end if
30: else if Type = Rspond and DestAddr = ID then
31:    broadcast < L, ID, Fwd, myPrntID, FTS, STS, myLevel >
32: else if Type = Fwd and DestAddr = ID and isFatherAtk = false then
33:    T8 ← CurrentTime()
34:    goto algorithm 4
35: else if Type = Corr and nodeType = PN and FID = myPrntID then
36:    logicaltime = logicaltime − diff
37: end if
```

The time synchronization process is started in Algorithm 2. The variable *myPrntID* is used to record father node's ID. When a *UN* receives an *Init* message (Line 7), it starts a timer *ST* (Line 8). Then it marks the source address *SrcID* of the message as *myPrntID* and sets *myLevel* to *Level* + 1 (Lines 9-10). If any node's *ST* timer expires (Line 1), it converts *nodeType* to *BN* (Line 2), and records the local timestamp $T1$ (Line 3). The function *CurrentTime*() is used to get the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

current local time. Then it replies a *Sync* message (Line 4). When a node receives a *Sync* message (Line 11), the node records the local time T2 (Line 12). If *myPrntID* is equal to FID of the message and the *nodeType* is UN (Line 13), the node becomes a *PN* and cancels timer ST (Lines 14-15). If the destination address of *Sync* message is equal to ID (Line 16), the node sends back an *Ack* message and records $T3$ (Lines 17-18). If a node receives an *Ack* message, it records the local time as $tempTime$ (Line 21). If ID is equal to FID of the message (Line 22), the node marks the $tempTime$ as $T6$ (Line 23). The *SrcID* in *Ack* message is marked as $tempID$ (Line 24). After that, it replies a *Rspond* message which carries variable $tempID$ at time $T7$ (Lines 25-26). If the node satisfies $DestAddr = ID$ and $SrcID = myPrntID$, then goto algorithm 3 (Line 28). If a node receives a *Rspond* message and *DestAddr* is equal to ID (Line 30), then it sends a Fwd message (Line 31). When a Fwd message is received, if *DestAddr* is equal to ID and *isFatherAtk* is $false$ (Line 32), it records the local time $T8$ and goes to algorithm 4 (Lines 33-34). If a *PN* receives a *Corr* message and FID is equal to *myPrntID* (Line 35), it uses $diff$ to adjust the clock (Line 36).

The time synchronization process of *PNs* and *BNs* in level one is shown in Algorithm 3. Variable $tempTime$ is marked as T4 (Line 1). Similarly, FTS and STS are marked as $T2$, $T3$ respectively (Lines 2-3). If the node satisfies *myLevel* is equal to 1 or *isFatherAtk* is true. (Line 4), it calculates the offset with four timestamps and compensates the $logicaltime$ (Lines 5-6). Then it broadcasts an *Init* message (Line 7). If the node is a *PN* (Line 8), it calculates the offset with two timestamps and compensates the $logicaltime$ (Lines 9-10).

Algorithm 3 Process of synchronizing *PNs* and *BNs* in level one

1: $T4 \leftarrow tempTime$
2: $T2 \leftarrow FTS$
3: $T3 \leftarrow STS$
4: **if** $myLevel = 1$ or $isFatherAtk = true$ **then**
5: $\quad offset = ((T2-T1)-(T4-T3))/2$
6: $\quad logicaltime = logicaltime + offset$
7: $\quad broadcast < -1, ID, Init, myPrntID, -1, -1, myLevel >$
8: **else if** $nodeType = PN$ **then**
9: $\quad offset = T2 - T3$
10: $\quad logicaltime = logicaltime + offset$
11: **end if**

Algorithm 4 shows the process of synchronizing *BNs*. First, FTS and STS are marked as $T5$, $T6$ respectively (Lines 1-2), and then the node calculates $diff$ (Line 3). If the $diff$ is less then the threshold value δ (Line 4), it compensates the $logicaltime$ with $offset$ according to Eq. 8 (Lines 5-6). Otherwise, It calculates $offset$ with Eq. 9 and compensates the $logicaltime$ (Lines 8-9). After that, it marks the father node as malicious node by setting variable *isFatherAtk* to true.

The space complexity of STSP is $\mathcal{O}(1)$, because each node stores a certain number of variables. Further, the time complexity of each node is $\mathcal{O}(1)$ in the initialization phase of STSP. In the time synchronization phase, nodes are divided into *PN*s and *BN*s. Each *BN* has a different number of *PN*s as child nodes. The time complexity of *PN*s is $\mathcal{O}(1)$ and the time complexity of *BN*s is $\mathcal{O}$(n). The largest time complexity in our algorithms is $\mathcal{O}$(n). Thus STSP is a lightweight time synchronization protocol.

Algorithm 4 Process of synchronizing *BNs*

1: $T5 \leftarrow FTS$
2: $T6 \leftarrow STS$
3: $diff \leftarrow ((T2+T3+T8)-(T4+T6+T7))/2$
4: **if** $diff \leqslant \lambda$ **then**
5: $\quad offset = ((T2-T1)-(T4-T3))/2$
6: $\quad logicaltime = logicaltime + offset$
7: **else**
8: $\quad offset = ((T6-T1)-(T8-T7))/2$
9: $\quad logicaltime = logicaltime + offset$
10: $\quad isFatherAtk \leftarrow true$
11: **end if**

V. Performance simulation

A. Simulation setup

We use the network simulation tool NS-2 to evaluate the secure performance of the STSP, TPSN [25] and STETS [28]. The secure performance and time consumption are compared in the following. We focus on the impact of the various number of malicious nodes. In order to reflect the security of the STSP, fake timestamps are broadcasted to influence the synchronization of the entire sensor network. For malicious nodes, the parameters α and β in the clock model are set to 1.0 and 0.001 respectively to form outliers. The percentage of false synchronization P is utilized to evaluate the security of STSP, which is calculated by Eq. 12. N_o stands for the total number of normal nodes. N_f represents the number of common nodes which synchronize to the wrong time.

$$P = \frac{N_f}{N_o} \tag{12}$$

We set up different numbers of malicious nodes in the fixed topology and the random topology. Then we show the security performance of STSP, which is obtained by running the simulations for 10000 times with different parameters. The communication range of each node is set to $100m$.

B. Overhead analyzation

In this section, we evaluate the overhead of STSP by calculating the number of exchanging messages. M is the total number of nodes. N_{TPSN} is used to represent the required messages to synchronize all nodes in TPSN. In TPSN, a spanning tree structure is constructed at first. During this process, each node needs to broadcast a message. Thus, the number of messages is M. Then it utilizes SRP model to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

synchronize the nodes along the edges in the spanning tree. The number of required messages is three to synchronize a node by using SRP model. Due to $M-1$ nodes need to be synchronized, the number of exchanging messages is $3(M-1)$. As a result, the number of required messages in TPSN is calculated in Eq.13.

$$N_{TPSN} = M + 3(M-1) = 4M - 3 \tag{13}$$

N_{STETS} is utilized to represent the required messages to synchronize all nodes in STETS. In the time synchronization process of STETS, nodes are divided into two types, which are *BN*s and *PN*s respectively. B is the number of *BN*s in spanning tree structure, thus the number of *PN*s is $M-B$. *BN*s utilize SRP model to synchronize the time. The number of required messages for *BN*s is $3(B-1)$. While *PN*s only receive messages and do not broadcast any messages. Thus the total number of required messages to synchronize all nodes is shown in Eq.14. Obviously, N_{STETS} is much smaller than N_{TPSN}.

$$N_{STETS} = 3(B-1) = 3B - 3 \tag{14}$$

N_{STSP} stands for the required messages to synchronize all nodes in STSP. O is the number of *BN*s whose level is equal to one. In STSP, the *BN*s whose level is one use SRP model to synchronize the time. While the *BN*s whose level is bigger than one use our proposed secure model. This secure model requires five messages to synchronize a node. Thus the total number of required messages in STSP is calculated in Eq. 15.

$$N_{STSP} = 3O + 5(B-O-1) = 5B - 2O - 5 \tag{15}$$

Compared with SRP model, two more messages are needed for the proposed secure time synchronization model to synchronize a node. Therefore the overhead of STSP is larger than STETS. According to Eq. 16, the difference number of required messages between STSP and STETS is Δ. However in dense wireless sensor networks, the number of *BN*s is limited. Thus, the value of Δ would not be too large and the overhead of STSP is acceptable for the dense networks.

$$\Delta = N_{STSP} - N_{STETS} = 2B - 2O - 2 \tag{16}$$

C. Security performance in fixed topology

64 nodes are deployed in the area of $500m$ * $500m$, which are divided into 8 rows and 8 columns. The distance between each row is set to $60m$. Similarly, the distance between each column is $60m$. The root node is in the bottom right corner of the topology. We evaluate the secure performance by setting up malicious nodes in different levels of the tree-based topology. Fig. 5 shows the impact of malicious nodes which located in level 1 to 8 of the spanning tree. With the level number increasing, more and more nodes are synchronized to the root node in TPSN and STETS. It can be seen that STSP can synchronize all common nodes to the root node. While TPSN and STETS can hardly synchronize any nodes to the root node when the malicious nodes are in level one. At the same time, we can see that STETS can synchronize more common nodes

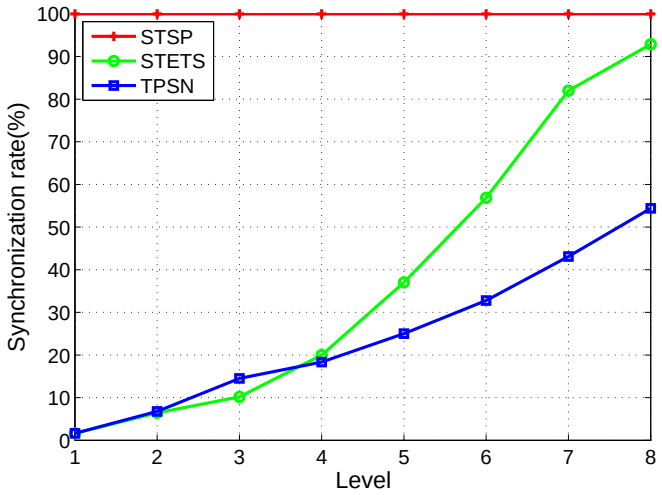


Fig. 5: Secure performance with the level of malicious nodes in spanning tree.

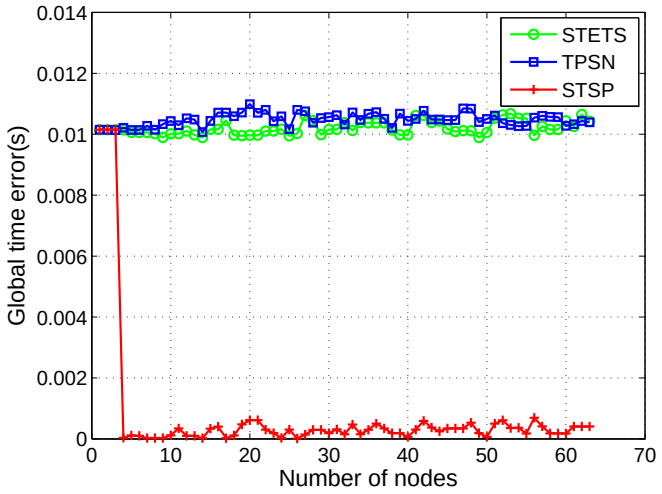


Fig. 6: Global time error when malicious nodes are located in level one.

to the root node compared with the TPSN when malicious nodes are over level four. When the malicious nodes are in the ninth level, over 90% nodes can synchronize to the time of root node, nevertheless TPSN only achieves 55%. Because TPSN only employs SRP model, while STETS combines SRP model and RRP model, and the nodes that employ RRP model limit the transmission of fake timestamps.

In the follows, we show the global time error of 64 nodes (including malicious nodes) under the impact of malicious nodes. The global time error represents the difference of calibration time between any node and its root node. For example in Fig. 6, Y-axis represents the global time error. X-axis stands for the number of nodes. When the malicious nodes are located in the first level of the spanning tree, all nodes in STETS and TPSN are synchronized to the false time. Therefore the global time error of TPSN and STETS is approximate to $0.01s$. However STSP synchronizes all normal nodes to root node and achieves high accuracy.

Fig. 7 shows the global time error when the malicious

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

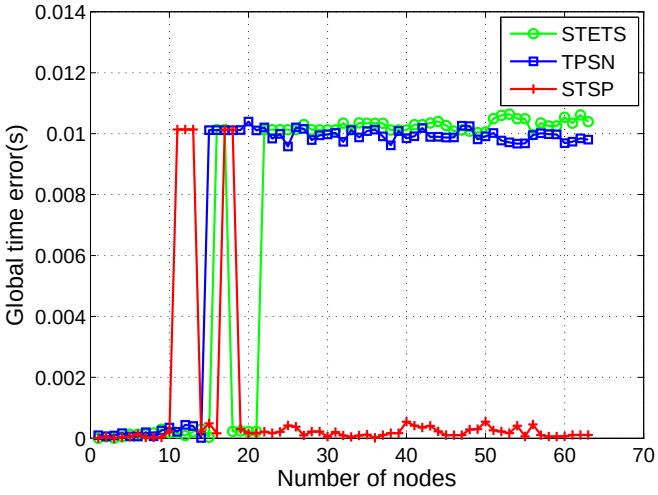


Fig. 7: Global time error when malicious nodes are located in level five.

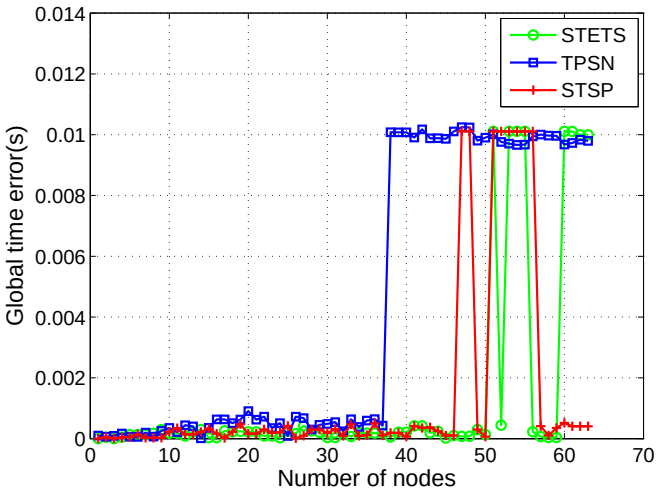


Fig. 8: Global time error when malicious nodes are located in level nine.

nodes are located in the level five. We can see that STSP can synchronize all the normal nodes to the root node. However TPSN and STETS algorithms only guarantee that the nodes whose level is less than five are synchronized to the root node, and the nodes after the fifth level are synchronized to the malicious nodes. It obviously that more nodes in TPSN are synchronized to false time compared with STETS. Similarly, Fig. 8 shows the situation that the malicious nodes are located in the ninth level. In conclusion, STSP is effective to prevent the nodes to utilize the fake timestamps that broadcasted from malicious nodes. Therefore STSP reduces the global time error of normal nodes in hostile environments.

D. Security performance in random topology

In this section, we evaluate the security performance of STSP with random topology. Fig. 9 shows the security performance when the rate of malicious nodes ranges from 0.1 to 0.7. The total number of nodes is set to 200. It can be concluded that the percentage of false synchronization increases with the rate of malicious nodes increasing. Even if the rate of malicious reaches up to 0.7, the percentage of false synchronization is only approximate to 5%, which is further lower than TPSN and STETS. STSP cannot guarantee all nodes synchronize to the root node due to the regardless of a situation, where multiple consecutive malicious *BN*s are deployed in the topology.

In Fig. 10, we evaluate the secure performance of three protocols in different scales of sensor networks. The X axis represents the total number of nodes, which consists of malicious nodes and normal nodes. It shows that when the proportion of malicious nodes in the network is fixed, the proportion of nodes affected by malicious nodes will not change obviously with the total number of nodes increasing, thus the secure performance of STSP is stabile, even if in large scale IoT.

In Fig. 11, we show the secure performance of 200 nodes. The proportion of malicious nodes increased from 10% to 70%. The percentage of false synchronization is approximate to 7% when 70% nodes are malicious. What's more, the false percentage doesn't increase significantly with the number of nodes increasing.

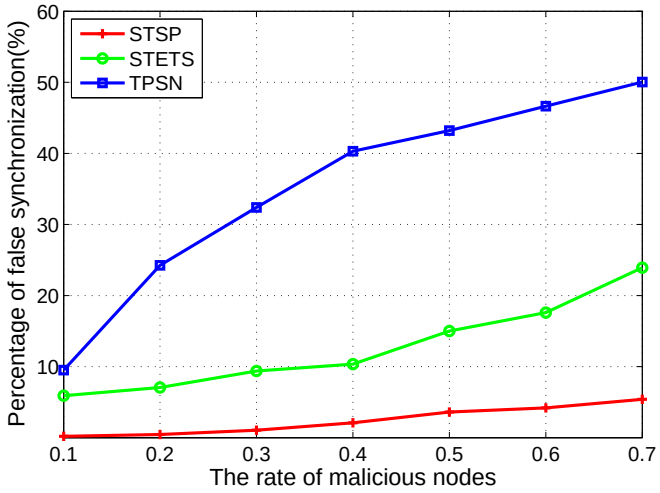


Fig. 9: Synchronization percentage with different percentage of malicious nodes.

E. Time consumption

In Fig. 12, we compare the time consumption of STSP with TPSN. Because the results of TPSN and STETS are same in linear topology, TPSN is chosen to represent the results. The topology of the node is a straight line, in which the distance between neighboring nodes is 80*m*. The time consumption of synchronization increases with the increasing of hop number. The time consumption of STSP is less than half of TPSN. Because STSP sends more messages when constructing a tree-based structure by utilizing the secure model. Due to the different uncertain elements like unstable time delay, the relationship of time consumption between STSP and TPSN is not a linear correlation. Although STSP consumes more time to synchronize a hop compared with TPSN, it greatly improved the security of time synchronization.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2714904, IEEE Internet of Things Journal

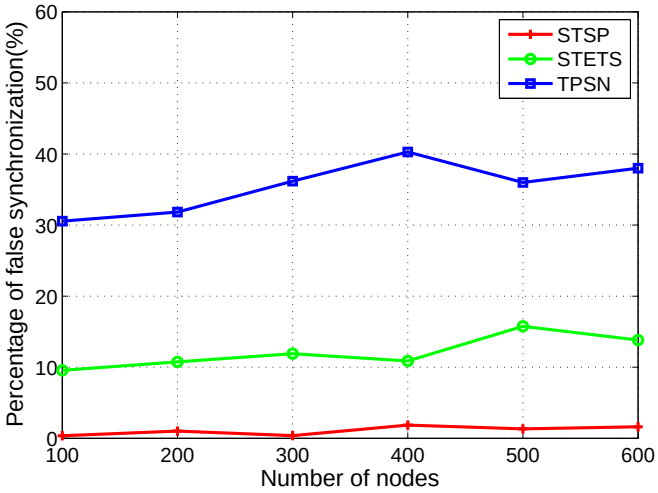


Fig. 10: Synchronization percentage with different sizes.

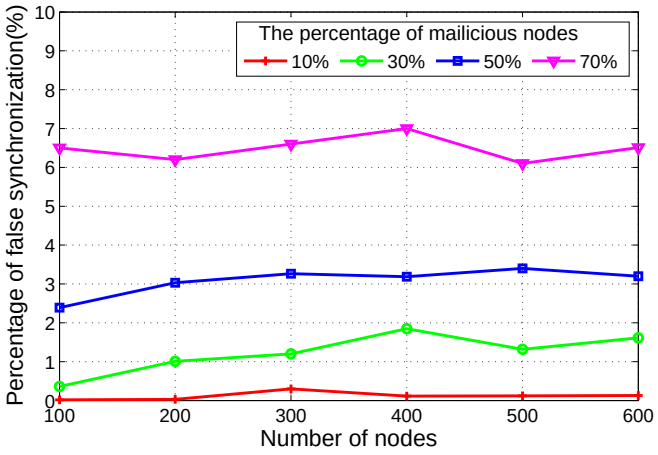


Fig. 11: Synchronization percentage with different sizes.

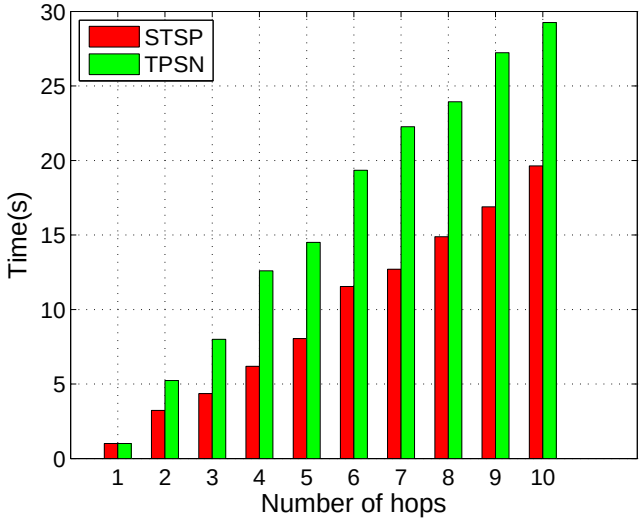


Fig. 12: Time consumption with hop distance.

VI. Conclusion

In this paper, we propose a model for making time synchronization against fake timestamps from malicious nodes in large scale IoT. A node uses its father node and grandfather node as reference nodes to detect the malicious nodes. Compared with the protocols employed SRP model, it can avoid normal nodes synchronizing to the malicious nodes. At the same time, we propose a secure time synchronization scheme STSP. Nodes establish a tree-based topology and use the secure model to synchronize time. In the simulation, we evaluate the secure performance compared with STETS and TPSN. The results reflect that STSP is effective to against fake timestamps. It performs steadily in large scale IoT. The further work will be focused on how to detect continuous multi-hop malicious nodes, to ensure that all nodes synchronized to the reference nodes. Besides, we will work on shorting the secure time synchronization period by sending fewer messages.

VII. Acknowledgment

This work is supported by National Natural Science Foundation of P.R. China (Grant No. 61672131, 61374154 and 61529101), NSF ECCS-1509212 and the Fundamental Research Funds for the Central Universities (Grant No. DUT16QY27).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] T. Qiu, Y. Lv, F. Xia, N. Chen, J. Wan, A. Tolba, "ERGID: An efficient routing protocol for emergency response Internet of Things," *Journal of Network and Computer Applications*, vol. 72, pp. 104–112, 2016.

[3] T. Qiu, A. Zhao, R. Ma, V. Chang, F. Liu, Z. Fu, "A Task-Efficient Sink Node Based on Embedded Multi-core SoC for Internet of Things," *Future Generation Computer Systems*, 2016. DOI: 10.1016/j.future.2016.12.024

[4] H. Song, D. Rawat, S. Jeschke, and C. Brecher, " Cyber-Physical Systems: Foundations, Principles and Applications," Boston, MA: Academic Press, 2016, pp. 1–514.

[5] G. Han, L. Wan, L. Shu, and N. Feng, "Two Novel DoA Estimation Approaches for Real Time Assistant Calibration System in Future Vehicle Industrial," *IEEE Sys. J.*, 2015, DOI: 10.1109/JSYST.2015.2434822.

[6] S. Jeschke, C. Brecher, H. Song, and D. Rawat, "Industrial Internet of Things: Foundations, Principles and Applications," Cham, Switzerland: Springer, 2017, pp. 1–715.

[7] E. Xu, Z. Ding, and S. Dasgupta, "Target tracking and mobile sensor navigation in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 177C-186, 2013.

[8] G. Han, J. Shen, L. Liu, and L. Shu, "BRTCO: A Novel Border Line Recognition and Tracking Algorithm for Continuous Objects in Wireless Sensor Networks," *IEEE Syst. J.*, 2016, DOI: 10.1109/JSYST.2016.2593949.

[9] G. Xu, W. Shen, and X. Wang, "Applications of wireless sensor networks in marine environment monitoring: A survey," *Sensors*, vol. 14, no. 9, pp. 16932–16954, 2014.

[10] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "An evolutionary multi-objective sleep scheduling scheme for differentiated coverage in wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1093–1102, 2012.

[11] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *IEEE Wireless Commun. Networking Conf. WCNC, New Orleans, LA, United states*, March 16-20, 2003, pp. 1266–1273.

[12] I. Skog and P. Händel, "Synchronization by two-way message exchanges: Cramer-rao bounds, approximate maximum likelihood, and offshore submarine positioning," *IEEE Trans Signal Process*, vol. 58, no. 4, pp. 2351–2362, 2010.

[13] A. Liu, Z. Zheng, C. Zhang, Z. Chen, and X. Shen, "Secure and energy-efficient disjoint multipath routing for wsns," *IEEE Trans. Veh. Technol.*, vol. 61, no. 7, pp. 3255–3265, 2012.

[14] J. Liu, Z. Zhou, Z. Peng, J. Cui, M. Zuba, and L. Fiondella, "Mobi-sync: efficient time synchronization for mobile underwater sensor networks," *IEEE Trans Parallel Distrib Syst*, vol. 24, no. 2, pp. 406–416, 2013.

[15] J. Granjal, E. Monteiro, and J. S. Silva, "Security in the integration of low-power wireless sensor networks with the internet: A survey," *Ad Hoc Networks*, vol. 24, pp.264–287, 2015.

[16] A. Boukerche, and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *IEEE Wireless Commun.*, vol. 14, no. 5, pp. 64–69, 2007.

[17] H. Song, G. A. Fink, and S. Jeschke, "Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications," Chichester, UK: Wiley-IEEE Press, 2017

[18] J. He, J. Chen, P. Cheng, and X. Cao, "Secure time synchronization in wirelesssensor networks: A maximum consensus-based approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 1055–1065, 2014.

[19] W. Dong and X. Liu, "Robust and Secure Time-Synchronization Against Sybil Attacks for Sensor Networks," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1482–1491, 2015.

[20] Y. Liu, J. Li, and M. Guizani, "Lightweight secure global time synchronization for wireless sensor networks," in *IEEE Wireless Commun. Networking Conf. WCNC, Paris, France*, April 1-April 4, 2012, pp. 2312–2317.

[21] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proc. Symp. Operating Systems Design and Implementation (OSDI), Boston, Massachusetts, United states*, December 09-11, 2002, pp. 147–163.

[22] D. Djenouri, "R4syn: Relative referenceless receiver/receiver time synchronization in wireless sensor networks," *IEEE Signal Process Lett*, vol. 19, no. 4, pp. 175–178, 2012.

[23] S. Jain and Y. Sharma, "Optimal performance reference broadcast synchronization (oprbs) for time synchronization in wireless sensor networks," in *Int. Conf. Comput., Commun. Electr. Technol., ICCCET, Maruthakulam, India*, March 18-19, 2011, pp. 171–175.

[24] K. L. Noh, Y. C. Wu, K. Qaraqe, and B. W. Suter, "Extension of pairwise broadcast clock synchronization for multicluster sensor networks," *Eurasip. J. Adv. Sign. Process, special issue on Distributed Signal Processing Techniques for Wireless Sensor Networks*, vol. 2008.

[25] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys Proc. First Int. Conf. Embedded Networked Sensor Syst., Los Angeles, CA, United states*, November 5-7, 2003, pp. 138–149.

[26] D. Djenouri, N. Merabtine, F. Z. Mekahlia, and M. Doudou, "Fast distributed multi-hop relative time synchronization protocol and estimators for wireless sensor networks," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2329–2344, 2013.

[27] L. He, "Time synchronization based on spanning tree for wireless sensor networks," in *Int. Conf. Wirel. Commun., Netw. Mob. Comput., WiCOM, Dalian, China*, October 12-14, 2008, pp. 1–4.

[28] T. Qiu, L. Chi, W. Guo, and Y. Zhang, "Stets: A novel energy-efficient time synchronization scheme based on embedded networking devices," *Microprocessors Microsyst*, vol. 39, no. 8, pp. 1285–1295, 2015.

[29] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Trans Parallel Distrib Syst*, vol. 25, no. 1, pp. 244–253, 2014.

[30] M. Akhlaq and T. R. Sheltami. 2013. "Rtsp: An accurate and energy-efficient protocol for clock synchronization in wsns," *IEEE Trans. Instrum. Meas*, vol. 62, no. 3, pp. 578–589, 2013

[31] Y. Kadowaki and H. Ishii, "Event-based distributed clock synchronization for wireless sensor networks," *IEEE Trans. Autom. Control.*, vol. 60, no. 8, pp. 2266–2271, 2015.

[32] L. Schenato and F. Fiorentin, "Average TimeSynch: A consensus-based protocol for clock synchronization in wireless sensor networks," *Automatica*, vol. 47, no. 9, pp. 1878–1886, 2011.

[33] J. He, P. Cheng, L. Shi, and J. Chen, "Time Synchronization in WSNs: A Maximum Value Based Consensus Approach," *IEEE Trans. Autom. Control*, vol. 59, no.3, pp. 660–675, 2014.

[34] 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, 2012.

[35] K. Sun, P. Ning, and C. Wang, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proc. 13th ACM Conference on Computer and Communications Security*, October 30-November 03, 2006, pp. 264–277.

[36] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," in *Ad hoc networks*, vol. 5, no. 1, pp 112–125, 2007.

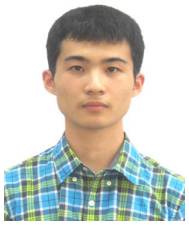
[37] D. J. Huang , W. C. Teng , K. T. Yang, " Secured flooding time synchronization protocol with moderator," *International Journal of Communication Systems*, vol. 26, no. 9, pp. 1092–1115, 2013.

[38] J. He, P. Cheng, L. Shi, J. Chen. SATS: Secure average-consensus-based time synchronization in wireless sensor networks. *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6387–6400, 2013

[39] X. Hu, T. Park, and K. G. Shin, "Attack-tolerant time-synchronization in wireless sensor networks," in *Proc. of IEEE INFOCOM*, Apr. 2008.

[40] S. Ganeriwal, S. Capkun, C. C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *WiSe 05: Proceedings of the 4th ACM workshop on Wireless security*, 2005.

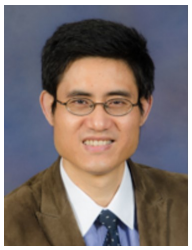
Tie Qiu (M'12-SM'16) received B.Sc from Inner Mongolia University of Technology, M.Sc and Ph.D from Dalian University of Technology (DUT) , China, in 2003, 2005 and 2012, respectively. He is currently Associate Professor at School of Software, Dalian University of Technology. He was a visiting professor at electrical and computer engineering at Iowa State University in U.S. (Jan. 2014-Jan. 2015). He serves as an Associate Editor of IEEE Access Journal, Computers and Electrical Engineering (Elsevier journal) and Human-centric Computing and Information Sciences (Springer Journal), an Editorial Board Member of Ad Hoc Networks (Elsevier journal) and International Journal on AdHoc Networking Systems, a Gest Editor of Future Generation Computer Systems (Elsevier journal). He serves as General Chair, PC Chair, Workshop Chair, Publicity Chair, Publication Chair or TPC Member of a number of conferences. He has authored/co-authored 8 books, over 60 scientific papers in international journals and conference proceedings. He has contributed to the development of 4 copyrighted software systems and invented 15 patents. He is a senior member of China Computer Federation (CCF) and a Senior Member of IEEE and ACM.



Xize Liu He is Master Student in School of Software, Dalian University of Technology (DUT), China. He is an excellent graduate student of DUT and has been awarded several scholarships in academic excellence and technology innovation. His research interests cover embedded system and internet of things .

Min Han (M'95-A'03-SM'06) received the B.S. and M.S. degrees from the Department of Electrical Engineering, Dalian University of Technology, Liaoning, China, in 1982 and 1993, respectively, and the M.S. and Ph.D. degrees from Kyushu University, Fukuoka, Japan, in 1996 and 1999, respectively. Since 2003, she is a Professor at Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology. She is the author of four books, more than 200 articles. Her current research interests are neural networks, chaos and their applications to control and identification.

Huansheng Ning (SM'13) received a B.S. degree from Anhui University in 1996 and Ph.D. degree in Beihang University in 2001. Now, he is a professor and vice dean of School of Computer and Communication Engineering, University of Science and Technology Beijing, China. His current research focuses on Internet of Things, cyber-physical modeling. He is the founder of Cyberspace and Cybermatics and Cyberspace International Science and Technology Cooperation Base. He serves as an associate editor of IEEE System Journal and IEEE Internet of Things Journal. He is the Co-Chair of IEEE Systems, Man, and Cybernetics Society Technical Committee on Cybermatics. He has hosted the 2013 World Cybermatics Congress (WCC2013/iThings2013/CPSCom2013/Greencom2013), and the 2015 Smart World Congress (Smart-World2015/UIC2015/ATC2015/ScalCom2015/CBDCom2015/IoP2015) as the joint executive chair. He gained the IEEE Computer Society Meritorious Service Award in 2013, IEEE Computer Society Golden Core Award in 2014.



Dapeng Oliver Wu (S'98-M'04-SM06-F'13) received B.E. in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 1990, M.E. in Electrical Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1997, and Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. He is a professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. He received University of Florida Term Professorship Award in 2017, University of Florida Research Foundation Professorship Award in 2009, AFOSR Young Investigator Program (YIP) Award in 2009, ONR Young Investigator Program (YIP) Award in 2008, NSF CAREER award in 2007, the IEEE Circuits and Systems for Video Technology (CSVT) Transactions Best Paper Award for Year 2001, and the Best Paper Awards in IEEE GLOBECOM 2011 and International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine) 2006. Currently, he serves as Editor in Chief of IEEE Transactions on Network Science and Engineering, and Associate Editor of IEEE Transactions on Communications, IEEE Transactions on Signal and Information Processing over Networks, and IEEE Signal Processing Magazine. He was the founding Editor-in-Chief of Journal of Advances in Multimedia between 2006 and 2008, and an Associate Editor for IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Wireless Communications and IEEE Transactions on Vehicular Technology. He is also a guest-editor for IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Cross-layer Optimized Wireless Multimedia Communications. He has served as Technical Program Committee (TPC) Chair for IEEE INFOCOM 2012, and TPC chair for IEEE International Conference on Communications (ICC 2008), Signal Processing for Communications Symposium, and as a member of executive committee and/or technical program committee of over 80 conferences. He was elected as a Distinguished Lecturer by IEEE Vehicular Technology Society in 2016. He has served as Chair for the Award Committee, and Chair of Mobile and wireless multimedia Interest Group (MobIG), Technical Committee on Multimedia Communications, IEEE Communications Society. He was an elected member of Multimedia Signal Processing Technical Committee, IEEE Signal Processing Society from Jan. 1, 2009 to Dec. 31, 2012. He is an IEEE Fellow.