Robust and Secure Time Synchronization against Sybil Attacks for Sensor Networks

Wei Dong, *IEEE Member*, and Xiaojin Liu

***Abstract*—Time synchronization is crucial for cyber-physical systems, e.g., wireless sensor and actuator networks. Cyber physical security is an important yet challenging problem. In particular, attacks to the time synchronization service may incur data distortion or even malfunction of the whole system. Sybil attack is one of the most common attack types for sensor networks, where a node illegitimately claims multiple identities. Existing secure time synchronization protocols, however, cannot well address Sybil attacks in the time synchronization process. We propose RTSP, a Robust and secure Time Synchronization Protocol to defend against Sybil attacks. Different from previous secure timesync protocols, RTSP employs a novel graph theoretical approach, thus is able to perform anomaly detection at the message level, instead of the node level. This fine-grained detection ability enables RTSP to become robust against Sybil attacks as well as node compromise and message manipulation attacks. Extensive experimental results show the effectiveness of our proposed protocol.**

I. Introduction

Time synchronization is crucial for cyber-physical systems (CPS) like wireless sensor and actuator networks where sensor nodes rely on a common time reference for event detection and industrial process control [1].

Many time synchronization algorithms and prototypes have been proposed in the literature, including Reference Broadcast Synchronization (RBS) [2], Flooding Time Synchronization Protocol (FTSP) [3], Timing-sync Protocol for Sensor Networks (TPSN) [4], Gradient Time Synchronization Protocol (GTSP) [5], and etc.

Recently, cyber physical security becomes increasingly important. This is especially true for sensor networks in CPS, where sensor nodes depend on wireless communications to collaborate with one another. Wireless communications are vulnerable to attacks such as congestion, eavesdropping, and manipulation.

In particular, attacks to the time synchronization service may incur data distortion or even malfunction of the whole system. For example, the recent IEEE 802.15.4e [6] standard requires all nodes to establish a common time for efficient medium access. In this case, the wireless communication may fail if nodes are unsynchronized due to attacks.

To defend such attacks, many secure time synchronization protocols are proposed [7], [8]. While we can borrow traditional security mechanisms, the design of secure time synchronization protocols still faces non-trivial challenges including:

- The resource constraints of sensor nodes preclude sophisticated mechanisms which require complex computations.
- Traditional centralized time synchronization protocols cannot be easily extended to adopt security mechanisms since the reliance on a few reference nodes often implies susceptibility to single points of failure, disrupting the synchronization service in the whole system.
- Sybil attack, where a single malicious node illegitimately presents multiple identities in the network, can invalidate the basic assumption that many secure time synchronization protocols build upon (i.e., a majority of nodes in the network are well behaving).

To address the first challenge, many secure timesync protocols identify possible attacks with a heuristic threshold of propagation delay or clock offset, avoiding complex authentication and encryption operations typically required for computer networks. To address the second challenge, a few secure timesync protocols build upon a distributed principle: a node adjusts its logical clock according to all the logical clocks of its neighbors, instead of receiving timesync messages from a distant reference node. The key idea for achieving security is to exploit the linearity of hardware clock readings to design checking mechanism. Unfortunately, existing secure timesync protocols are not well designed to cope with Sybil attacks.

Sybil attack is a particularly harmful attack against sensor networks, where a node illegitimately claims multiple identities. Newsome *et al.* demonstrate that the attack can be exceedingly detrimental to many important functions of the sensor network such as routing, resource allocation, misbehavior detection, etc [9]. For example, data aggregation is a widely used technique in sensor networks in order to conserve energy. A small number of malicious nodes reporting incorrect sensor readings might be unable to significantly affect the computed aggregate. However, by using the Sybil attack, one malicious node may be able to contribute to the aggregate many times with different identities, thus altering aggregate reading. Sybil attack is also an important threat for time synchronization. If a single Sybil node reports incorrect timestamps with different identities to neighbors, its neighbors may never converge to the correct global time with existing distributed time synchronization protocols. For example, Attack-tolerant Time-Synchronization Protocol (ATSP) [7] and Secured Maximum consensus based Time Synchronization (SMTS) [8] can detect malicious nodes and discard the timestamps sent from them. However, a single node can control a substantial fraction of

The authors are with the College of Computer Science, Zhejiang University. Email: dongw@zju.edu.cn.

the ID space under Sybil attack, causing ATSP and SMTS to detect too many malicious nodes. Hence, many sensor nodes cannot synchronize their logical clocks since they cannot find legitimate nodes in their neighborhood.

To address these challenges, we propose RTSP, a Robust and secure Time Synchronization Protocol to specifically deal with Sybil attack which is not well addressed by existing protocols [7], [8]. The key idea of RTSP is to employ a graph theoretical approach to perform anomaly detection at the message level, instead of the node level. RTSP can differentiate valid timestamps from invalid timestamps piggybacked in the synchronization messages. Although Sybil attacks could cause a portion of invalid timestamps for many nodes, they can be accurately detected and safely discarded. RTSP builds upon a novel graph algorithm to distinguish valid timestamps from invalid timestamps, i.e., RTSP detects anomaly at the timestamp level, instead of the node level. This fine-grained detection ability enables RTSP to become robust against Sybil attacks as well as manipulation attacks.

The contributions of this paper are summarized as follows:

- We propose a novel graph theoretical approach to detecting timestamp anomalies at a fine-grained granularity.
- We design and implement RTSP which specially deals with Sybil attacks. We implement RTSP and evaluate its effectiveness by extensive simulations.

The rest of this paper is structured as follows. Section II introduces the large body of related work. Section III describes the system model. Section IV presents the design of RTSP. Section V shows the evaluation results, and finally, Section VI concludes this paper and gives future research directions.

II. Related Work

Time synchronization has been studied extensively in the literature. Traditional timesync algorithms (e.g., the Network Time Protocol (NTP)) are not suitable for sensor networks due to their complexity. We divide existing work into three categories: centralized synchronization which relies on a synchronization root, distributed synchronization, and secure synchronization protocols.

A. Centralized Synchronization

RBS [2] is a scheme in which nodes send reference beacons to their neighbors using physical-layer broadcasts. A reference broadcast does not contain an explicit timestamp; instead, receivers use its arrival time as a point of reference. Hence, the sender's nondeterminism can be removed from the critical path in the synchronization process. RBS is designed for single-hop time synchronization only. For multihop networks, nodes which participate in more than one cluster can be employed to convert the timestamps between local clock values of different clusters.

TPSN [4] provides network-wide time synchronization in a sensor network. The algorithm works in two steps. In the first step, a hierarchical structure is established in the network and then a pair wise synchronization is performed along the edges of this structure to establish a global timescale throughout the network. Eventually all nodes in the network synchronize their clocks to a reference node.

FTSP [3] also provides network-wide synchronization by electing a synchronization root in the whole network. The FTSP achieves its robustness by utilizing periodic flooding of synchronization messages, and implicit dynamic topology update. The unique high precision performance is reached by utilizing MAC-layer timestamping and comprehensive error compensation including clock skew estimation.

Many recent works improve the synchronization accuracy by fast propagating timesync messages, (such as PulseSync [10] and Glossy [11]), or adaptively changing timesync intervals [1].

RTSP builds on top of distributed synchronization protocols. Distributed protocols are more secure than centralized protocols since the failure of the reference node will not disrupt the synchronization service. The RTSP security mechanism cannot be directly applied to centralized protocols. It would be interesting to devise a secure mechanism against Sybil attacks for existing centralized protocols.

B. Distributed Synchronization

A completely distributed synchronization algorithm was proposed in [12]. In Reachback Firefly Algorithm (RFA), each node periodically generates a pulse (message) and observes pulses from other nodes to adjust its own firing phase. RFA only provides synchronicity, nodes agree on the firing phases but do not have a common notion of time.

These shortcomings are tackled by GTSP [5] which is designed to provide accurately synchronized clocks between neighbors. GTSP works in a completely decentralized fashion: each node periodically broadcasts its time information. Synchronization messages received from direct neighbors are used to calibrate the logical clock. The algorithm requires neither a tree topology nor a reference node, which makes it robust against link and node failures.

Schenato and Fiorentin [13] propose an average time-sync (ATS) protocol, which consists of two averaging consensus algorithms. Nevertheless, it generally requires a large amount of data exchanges and the converging speed may be quite slow when the network size grows large. To this end, a maximum consensus based time synchronization protocol (MTS) is proposed in [14]. It has been shown that MTS converges faster than ATS. Meanwhile, these consensus based protocols are able to compensate both clock skew and offset simultaneously.

Distributed synchronization is more robust than centralized synchronization since a single node failure will not affect the synchronization process. Our RTSP security mechanism can be applied to other distributed synchronization protocols, e.g., ATS [13], GTSP [5].

C. Secure Synchronization

Ganeriwal *et al.* [15] first studied the problem of time synchronization in the presence of malicious nodes. They outlined a few attacks on existing time synchronization schemes, including the pulse delay attack, where an adversary deliberately delays the transmission of synchronization messages in order to magnify the time offset between it and the neighboring nodes. The authors also proposed secure single-hop, multi-hop and group time synchronization protocols for wireless sensor networks. The work in [15] focuses on message manipulation and pulse-delay attacks while our current work can defend against Sybil attacks as well as node compromise and message manipulation attacks.

Sun *et al.* [16] proposed secure and resilient pairwise and global time synchronization protocols that use authenticated MAC layer time-stamping and the μTESLA broadcast authentication protocol to overcome attacks by malicious motes. Rahman *et al.* propose a protocol in [17], which uses pairing and identity-based cryptography to secure the time synchronization to reduce the communication and storage requirements of each node. The above works build on top of authentication mechanisms while our current work builds on top of statistical mechanisms. Authentication requires key management and incurs more computation overhead. Moreover, these approaches cannot defend against Sybil attacks.

Hu *et al.* propose a distributed and secure synchronization protocol ATSP [7] that can tolerate attacks of node compromising, packet faking and delaying. The key idea is to exploit the linearity of hardware clock readings to design checking mechanism. ATSP exploits the high temporal correlation existing among adjacent nodes in a WSN to detect falsified clock values advertised by attackers or compromised nodes.

ATSP only checks the correctness of the offset value. SMTS [8] is concerned with secure time synchronization with the clock model which requires both skew and offset compensation under message manipulation attacks. A novel secured maximum consensus based time synchronization (SMTS) protocol is proposed to detect and invalidate message manipulation attacks. Different from existing secure synchronization protocols, our protocol cannot only detect malicious nodes under manipulation attacks but also can detect individual invalid timestamps by using a novel graph theoretical algorithm, making it robust against Sybil attacks.

III. System Models

A. Network Model

We consider a wireless network consisting of a large number of resource-constrained sensor nodes. There is neither a reference node nor a root node, i.e., we rely on distributed algorithms for time synchronization. Each node has a sufficient number of neighboring nodes to accelerate the synchronization process. Each node relies on a broadcast service to announce its logical clock to its neighboring nodes.

B. Attack Model

Since wireless sensor networks usually operate in an unattended environment, they are subject to a variety of attacks. Our paper focuses on Sybil attacks targeting the timesync service. Under *Sybil attack*, a single malicious node illegitimately presents multiple identities in the network and announces incorrect synchronization messages for those nodes under attack.

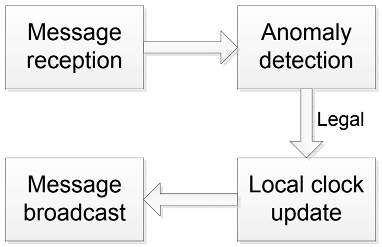

Fig. 1: RTSP synchronization process for a node.

C. Clock Model

Each sensor node is equipped with external crystal oscillators which are used as clock source for a counter register of the microcontroller. These oscillators exhibit drift which is only gradually changing depending on the environmental conditions such as ambient temperature or battery voltage and on oscillator aging. For a relatively extended period of time, the clock can be approximated with good accuracy by an oscillator of fixed frequency. The *local hardware clock* of a node can thus be approximated as $\tau = (1+\rho)t + \Delta$ where t is the reference clock, ρ is the clock drift and Δ is the clock offset. The clock drift can usually be bounded for a specific platform. For example, the TelosB datasheet reports a maximum drift of 40ppm with respect to the reference clock.

The value of the local hardware should not be adjusted manually since other hardware components may depend on the continuously running hardware clock. The *logical clock value* is computed as a function of the hardware clock:

$$L(t) = \alpha\tau(t) + \beta \tag{1}$$

By adjusting the coefficients of the function, the logical clock value represents the synchronized time of a node.

IV. RTSP Design

A. Overview

RTSP is based on distributed algorithm to achieve synchronization, i.e., nodes adjust its logical clock based on the timestamps advertised by all its neighboring nodes. Such a distributed mechanism makes timesync much more robust since the failure of the reference node will not disrupt the synchronization service.

From the perspective of a node, the synchronization process consists of four main steps shown in Fig. 1.

The message reception, clock update, and message broadcast process are common in time synchronization protocol. Currently, we implement RTSP as a full protocol. The security mechanism in RTSP, however, can be applied to other distributed synchronization protocols: when a sufficient number of synchronization messages are received, a node can employ the algorithm proposed in Section IV-C to filter out incorrect messages, thus can defend against Sybil attacks as well as node compromise and message manipulation attacks. Distributed protocols are more secure than centralized protocols since the failure of the reference node will not disrupt the synchronization service.

We will describe the timesync protocol in Section IV-B. Our key novelty is the proposition of the anomaly detection algorithm which can detect valid/invalid timestamps at the per-message level. The detection algorithm will be described in Section IV-C. Section IV-D presents a security analysis on RTSP. Section IV-E discusses how RTSP can be implemented in the IEEE 802.15.4e [6] standard and IETF RPL [18] protocol.

B. Time Synchronization

RTSP is based on distributed synchronization algorithms such as ATS [13] and MTS [14] to achieve network-wide synchronization. We adopt a distributed algorithm similar to MTS since MTS achieves faster convergence than previous distributed algorithms.

We adopt MAC-layer timestamping to remove sender side and receiver side uncertainties. At a high level, the receiver receives a synchronization message containing the sender's sending time at the MAC layer t_S. The receiver can also obtain the receiving time at the MAC layer t_R. We note that t_S is measured using the sender's local clock and t_R is measured using the receiver's local clock. t_S and t_R correspond to the sending and the receiving events at the MAC layer. In RTSP's synchronization process, we consider that t_S and t_R correspond to the same physical event. We can ignore the communication delay since the send timestamp is taken when the preamble of the packet is transmitted and the receive timestamp is taken when the preamble of the packet is received. Since these timestamps are taken in the SFD (start frame delimiter) interrupt, sender side uncertainties and receive side uncertainties can be removed. The in-air transmission time can also be safely ignored since the radio transmission time over 802.15.4 links (typically less than 100m) is negligible. We use t_E to denote the reference time for this event.

Let's denote $\tau_i(t)$ as the local clock of a node i, and $L_i(t)$ as the logical clock. Initially,

$$L_i(t) = \alpha_i \tau_i(t) + \beta_i \tag{2}$$

Each node will announce both α_i and β_i in the timesync beacon messages. The reception of one timesync beacon (at node i from node j) gives the logical clock skew α_j and the logical clock offset β_j for neighbor j. The timesync beacon also corresponds to one clock offset between the sender and receiver: $\Delta_{ij} = t_j^S - t_i^R$. The reception of two synchronization beacons correspond to an additional clock drift of the sender with respect to the receiver: $\rho_{ij} = \frac{t_j^S - t'^S_j}{t_i^R - t'^R_i} - 1$ where t'^S_j and t'^R_i denote the sending and receiving times of the previous synchronization message. Assuming there are n receivers in the sender's neighborhood, the sender can obtain the following sets: $\{\alpha_j\}_{j=1}^n$, $\{\beta_j\}_{j=1}^n$, and $\{\rho_{ij}\}_{j=1}^n$.

The receiver then calculates the following quantities: $\alpha_{ij} = (\rho_{ij}+1)\alpha_j$, $\beta_{ij} = L_j(t_E) - \alpha_{ij}\tau_i(t_E) = \alpha_j t_j^S + \beta_j - \alpha_{ij} t_i^R$. We then determine the k-th neighbor which has the maximum clock skew: $k = \arg\min_k(\alpha_{i1}, \alpha_{i2}, ..., \alpha_{in})$. α_i and β_i are updated as follows:

$$\begin{aligned}\alpha_i &= \alpha_{ik}\\ \beta_i &= \beta_{ik}\end{aligned} \tag{3}$$

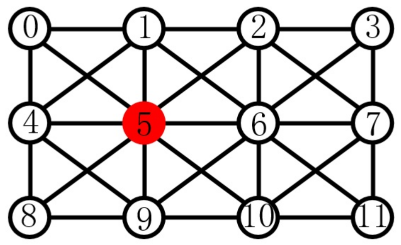

Fig. 2: A grid network topology consisting of 12 sensor nodes.

Finally, node i updates its logical clock according to Eq. (2).

It has been shown in [14] that all logical clocks will eventually converge to a common time. However, the above algorithm is designed for a benign environment in which every sensor node behaves normally and follows the underlying protocol faithfully. The performance will severely degrade if the sensor network is under security attacks.

Under manipulation attack. An attacker can advertise falsified sending times to potential receivers. Without proper designs, the potential receivers may adjust their clocks to have a large deviation from its desired value, making it impossible, or at least take a very long time, for all sensor nodes' clocks to converge. Hence, each node must verify the trustworthiness of synchronization messages it receives from its neighbors before using them to adjust its own clock values. Many previous works employ mechanisms in order to accurately detect these abnormal nodes. The key idea for achieving security is to exploit the linearity of hardware clock readings to design checking mechanism. For example, ATSP [7] exploits the linearity of clock offsets for a given neighbor to detect abnormal nodes. SMTS [8] additionally checks the correctness of clock drifts in addition to the clock offsets. Both algorithms detect compromised nodes after receiving a given number of synchronization messages: if the timestamps in those messages are detected to be inconsistent with each other, the node which these messages originate is suspected to have been compromised. Other nodes may blacklist and/or reject announcements from a neighbor if it behaved anomalously more than a predefined number of times.

Under Sybil attack. The security mechanisms employed in ATSP and SMTS are effective in defending against message manipulation attacks. They are, however, ineffective in the face of sybil attacks where a single malicious node illegitimately presents multiple identities in the network and announces incorrect synchronization messages for those nodes under attack. In this case, a node may regard many nodes under attack in its neighborhood as malicious nodes, thus losing synchronization with the rest of the network.

In order to show how existing algorithms perform under Sybil attack, we conduct a TOSSIM simulation on a grid network with 12 nodes (shown in Fig. 2). Suppose at the first stage, all the nodes behave exactly according to the our timesync protocol. However, node 5 is compromised by the attacker and will broadcast $s_5 + w_5$ every 5 synchronization intervals to its neighbors (where s_5 is the actual sending time and w_5 is randomly chosen from 5s–10s), illegitimately

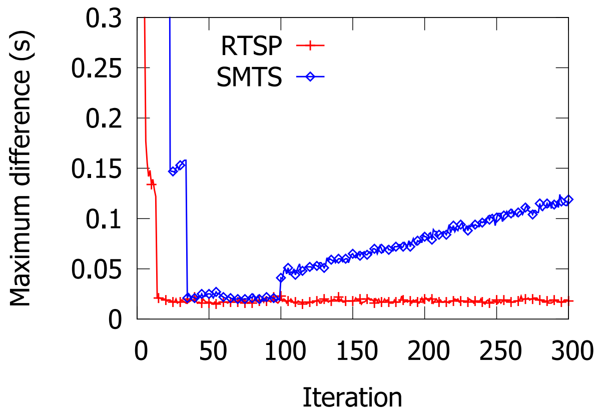

Fig. 3: Performance of SMTS under Sybil attack.

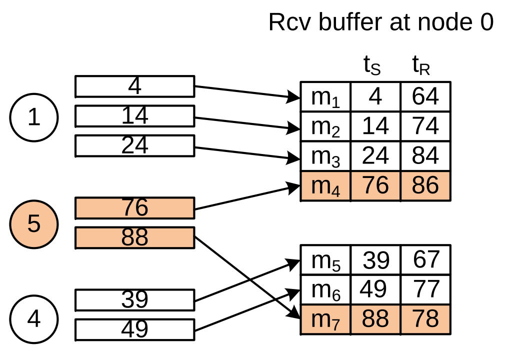

Fig. 4: An illustrative example to show how Algorithm 1 works. The shaded messages indicate incorrect messages. t_S denotes the send timestamp and t_R denotes the receive timestamp.

presenting multiple identities, i.e., node IDs 0, 1, 5, 8, 9 which node 5 can hear from. In this case, node 4 regards all its neighbors as malicious nodes by employing SMTS [8], making node 4's clock deviate significantly from the rest of the network. Fig. 3 shows the timesync accuracy of node 4. We can see that node 4 cannot synchronize with the rest of the network under Sybil attack.

C. The Detection Algorithm

We would like to devise a robust algorithm to defend against the abovementioned Sybil attacks. We assume the attacker only advertises the incorrect synchronization message *infrequently*. Otherwise, it can be easily detected since the number of synchronization message significantly deviates from the normal behavior. One might argue that any Sybil attack in this kind can be detected by enforcing a strict time synchronization interval, say 10 minutes. Hence, abnormal behavior violating this rule can be caught. However, most timesync algorithm adopt adaptive timesync intervals to strike a reasonable balance the synchronization accuracy and overhead [1].

The key insight of our approach is to exploit the conformance relationship between legal timestamp so that illegal timestamps can be detected and discarded. Before introducing the algorithm, we first illustrate how our algorithm works through an example.

Illustrative Example. We use an example to illustrate how RTSP works. Consider four nodes 0, 1, 4, 5 in Fig. 2 where node 5 is the Sybil node. We consider how node 5 can affect node 0's synchronization process.

Under the normal condition, node 0 would receive messages from nodes 1 and 4 with correct send timestamps. Fig. 4 shows that node 1 sends three messages to node 0 with a period of 10 (time units). The send timestamps are 4, 14 and 24, respectively, in node 1's local clock. Similarly, node 4 sends two messages to node 0 with send timestamps 39 and 49, respectively, in node 4's local clock. When node 0 receives these messages, it records the receive timestamps. For messages sent from node 1, the receive timestamps are 64, 74 and 84, respectively. For messages sent from node 4, the receive timestamps are 67 and 77, respectively.

The key insight of our design is that the difference of send timestamps and difference of the receive timestamps should be roughly the same for these successive messages from a node, e.g., $14-4=74-64$ for messages m_1 and m_2, and $24-14=84-74$ for messages m_2 and m_3. This relation is more precisely described by Eq. (4).

We now consider the case when node 5 attacks the timesync process and it advertises incorrect send timestamps. For example, node 5 incorrect messages, illegitimately claiming the identity of node 1 and node 4. Existing secure timesync protocol would regard node 1 as a malicious node since the four messages from node 1 (including the incorrect message sent by node 5) do not obey the linear clock model. Similarly, node 0 would regard node 4 as a malicious node. Therefore, node 0 is "disconnected" from the rest of the network, and remains unsynchronized when node 5 continuously sends incorrect messages.

Our RTSP security mechanism, on the contrary, can detect the incorrect message from node 5 since it does not "conform" to the other correct messages, e.g., $76-4 \neq 86-64$ for messages m_1 and m_4, $76-14 \neq 86-74$ for messages m_2 and m_4. When the incorrect messages (i.e., m_4 and m_7 in this example) can be filtered out, node 0 can remain synchronized with node 1 and node 4. However, there are challenges in automatically detecting such incorrect messages.

We now formally introduce our approach. Assume the buffer size is M for each neighbor, storing the most recent M timesync messages for anomaly detection. We introduce a conformance relationship between any two timesync messages, m_i and m_j ($i<j$ and j is the more recent message), in the buffer.

Definition 1: Two timesync messages from a given neighbor are said to be *conforming*, denoted as $m_i \sim m_j$, iff

$$1-\rho_m \leq \frac{m_j.t_S - m_i.t_S}{m_j.t_R - m_i.t_R} \leq 1+\rho_m \quad (4)$$

where $m_k.t_S$ and $m_k.t_R$ denote the corresponding sending timestamp and receiving timestamp ($k=i,j$) and ρ_m denotes the maximum clock drift between two nodes.

We note that the conformance relationship is a *necessary* condition for two correct messages since the clock drifts for any two nodes are bounded. For example, the maximum clock drift $\rho_m = 40$ ppm for the TelosB sensor node platform.

We use a graph $G = (V,E)$ to represent the conforming relationship between two messages with each message m_i corresponding to a vertex $v_i \in V$, i.e., $e = (v_i, v_j) \in E$ iff $v_i \sim v_j$. Consider well-behaved neighbor, all its messages are conforming to one another, i.e., the graph G is a clique.

If there are random errors, the graph G is no longer a clique. It is reasonable to consider that all the correct messages form the maximum clique in the graph when the injected errors are infrequent. Therefore, detecting the abnormal timestamps translates to detecting the maximum clique in the graph: the vertices not in the maximum clique are considered anomalies.

It is, however, NP-hard to detect the maximum clique in a general graph. It is hence desirable to devise efficient algorithms for solving this problem. Fortunately, we find that the graph has certain properties that make the maximum clique identification problem tractable. In particular, we can obtain the following theorem:

Theorem 1: If m_i and m_j are conforming, and m_j and m_k are conforming ($i < j < k$, i.e., m_i,m_j,m_k are in increasing order of time), m_i and m_k are also conforming, i.e., $m_i \sim m_j$ and $m_j \sim m_k \Rightarrow m_i \sim m_k$.

Remarks: For the illustrative example shown in Fig. 4, we can conclude that m_1 and m_3 are conforming according to Theorem 1. This is because m_1 and m_2 are conforming ($14-4 = 74-64$), and m_2 and m_3 are also conforming ($24-14 = 84-74$). This theorem shows that the conformance relationship is *transitive*.

Corollary 1: $m_{i_1} \sim m_{i_2}$, $m_{i_2} \sim m_{i_3}$, ..., $m_{i_{n-1}} \sim m_{i_n} \Rightarrow m_{i_1} \sim m_{i_n}$ ($i_1 < i_2 < ... < i_n$).

Let's revise graph $G = (V,E)$ to $\overrightarrow{G} = (V, \overrightarrow{E})$ by considering edge directions: $(v_i, v_j) \in \overrightarrow{E}$ iff $v_i \sim v_j$ and $i < j$ (i.e., m_j is the more recent message). We note that in $\overrightarrow{G}$, edge always points from an old message to a new message. Theorem 1 implies that graph $\overrightarrow{G}$ is *transitive*. Maximum clique identification in directed transitive graph is known to have efficient algorithms. We employ a polynomial algorithm with $O(M^2)$ time complexity in Algorithm 1.

This algorithm uses a dynamic programming approach to find the maximum clique. Let $\overrightarrow{G}_i$ be a subgraph defined on a subset of vertices in $V_i = \{v_1, ..., v_i\}$ and c_i denotes the size of the maximum clique in $\overrightarrow{G}_i$. In general, when $c_1, c_2, ..., c_{i-1}$ are known, we look for vertices among $v_1, v_2, ..., v_{i-1}$ which are connected by an edge to v_i. If there are no such vertices, $c_i = 1$. Otherwise,

$$c_i = 1 + \max_{1 \le p < i \text{ and } (v_p, v_i) \in \overrightarrow{E}} \{c_p\} \tag{5}$$

Finally, the size of the maximum clique is $\max_{1 \le i \le n} c_i$. The array x[] is for a backtrace operation for enumerating all nodes in the clique.

Let's examine an example shown in Fig. 5 to illustrate how Algorithm 1 works. In Fig. 5, each vertex represents a message from a given source node. There is an edge between two messages if they conform to each other. In Fig. 5, v_1, v_2, v_4, and v_5 form a clique of size 4. v_1, v_2, and v_3 form a clique with size 3. We see that the maximum clique size is 4 for this simple example.

Algorithm 1 Maximum Clique Identification

```
Input: $\overrightarrow{G}(V, \overrightarrow{E})$ where $V = \{v_1, ..., v_n\}$
Output: $K \subseteq V$: set of vertices in the maximum clique of $\overrightarrow{G}$
 1: Define $\overrightarrow{G}_i$ as a subgraph of $\overrightarrow{G}$ with vertex set $V_i = \{v_1, ..., v_i\}$.
 2: Define c[i] as the size of the maximum clique containing $v_i$ in $\overrightarrow{G}_i$.
 3: // Initialize arrays c[] and x[] where x[] is for backtrace purpose.
 4: for $i$: $1 \le i \le n$ do
 5:     $c[i] \leftarrow 1$ and $x[i] \leftarrow 0$
 6: end for
 7: for $i$: $2 \le i \le n$ do
 8:     for $p$: $1 \le p < i$ do
 9:         if $(v_p, v_i) \in \overrightarrow{E}$ and $1 + c[p] > c[i]$ then
10:             $c[i] \leftarrow c[p] + 1$ and $x[i] \leftarrow p$
11:         end if
12:     end for
13: end for
14: $m \leftarrow \arg\max_{1 \le i \le n} \{c[i]\}$.
15: while $m$ do
16:     $K \leftarrow K \cup \{v_m\}$
17:     $m \leftarrow x[m]$
18: end while
```

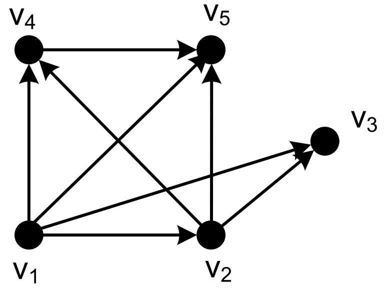

Fig. 5: The graph representation of the received timesync messages. Each vertex represents a received timesync message at the receiver. Correct messages (i.e., with correct timestamps) must conform to each other. There is an edge between two vertices (messages) if they conform to each other.

Our algorithm aims to automatically find the maximum clique since each message in the maximum clique can be regarded as correct. Maximum clique identification in a general graph is known to be NP-complete. Thanks to the transitive property of the graph, we are able to devise a polynomial time algorithm (Algorithm 1) for identifying the maximum clique.

We use the example shown in Fig. 5 to illustrate the working details of Algorithm 1.

1) When $i = 2$: $p = 1$, $1 + c[1] = 2$. So $c[2] = 2$, $x[2] = 1$.
2) When $i = 3$: if $p = 1$, $1 + c[1] = 2$; if $p = 2$, $1 + c[2] = 3 > 2$. So $c[3] = 3$, $x[3] = 2$.
3) When $i = 4$: if $p = 1$, $1 + c[1] = 2$; if $p = 2$, $1 + c[2] = 3 >$

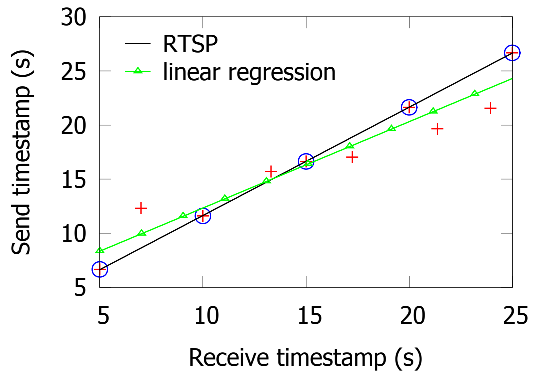

Fig. 6: Illustration of our approach

2; and there is no edge between v_3 and v_4. So $c[4] = 3$, $x[4] = 2$.

4) When $i = 5$: $1 + c[1] = 2$; $1 + c[2] = 3$; $1 + c[4] = 4$. So $c[5] = 4$ and $x[5] = 4$.

We see that Algorithm 1 indeed finds the maximum clique with size 4 in this example.

We can formally prove that Algorithm 1 actually finds the maximum clique in the graph.

Theorem 2: Algorithm 1 outputs all nodes in the maximum clique in the graph.

Till now, we have described an efficient algorithm for identifying the maximum clique in a graph, i.e., the correct messages from a neighbor. Hence, a receiver can discard those incorrect ones to defend against possible attacks.

We additionally enhance our detection algorithm by adopting two mechanisms. First, we also exploit linearity to devise checking mechanisms similar to [7], [8] after discarding the incorrect timestamps already detected. This mechanism further filters out possible small noises. Second, we check consistency among all neighbors after the timesync process converges. This mechanism will help detect a few malicious nodes which keep misbehaving, i.e., announcing false but conforming timestamps deliberately.

Figure 6 illustrates how our approach works. Each dot represents a received message with a send timestamp and a receive timestamp. The dots with circles represent correct messages. Our approach can correctly detect these correct messages since they are conforming to each other and constitute the maximum clique in the graph consisting of 10 vertices. As a comparison, we consider the linear regression method adopted in previous works. Without detecting the incorrect messages in the first place, the fitted line (denoted as “linear regression”) would probably deviate significantly from the ground truth (denoted as “RTSP”) because of large errors in the incorrect messages. Error detection based on the fitted line can thus cause errors in the detection results. For example, the rightmost dot would be probably identified as “incorrect” since it has a large distance to the fitted line.

D. Security Analysis

The goal of attackers is to influence and cause a significant bias in the synchronization process by providing false clock information. RTSP can defend against both node compromise and message manipulation attacks.

A *node compromise* [19] often consists of three stages: physically obtaining and compromising the sensors, redeploying the compromised sensors, and compromised nodes launching attacks after their rejoining the network. The adversary may compromise some nodes, and exploit the compromised nodes in arbitrary ways to attack time synchronization. For example, the adversary may instruct the compromised nodes to (selectively) delay or drop time synchronization messages. The adversary may also instruct the compromised nodes not to cooperate with others, and inject false time synchronization messages. Under *Message manipulation* attack, an attacker can modify or fake synchronization messages, causing other nodes to adjust to an incorrect time.

Both node compromise and message manipulation is to change the sending timestamps so that the acceptance of these falsified timestamps by victim nodes will lead to incorrect adjustment of their logical clocks. It is worth noting *message delay*, another category of attack, is equivalent to modifying the receive timestamps. Like ATSP [7] and SMTS [8], RTSP is resilient to these attacks since large errors will be detected by our first-stage graph theoretical approach and small noises will be filtered out by our second-stage checking mechanism. This forces the attackers to weaken the attack strength by announcing conforming timestamps. However, as long as malicious nodes do not form a majority within the local region of interest, such attacks can be detected by checking consistency among all neighboring nodes.

Different from ATSP and SMTS, RTSP is resilient to Sybil attacks. This is because RTSP employs a finer-grained detection algorithm to differentiate correct/incorrect timestamps. Hence the falsified timestamps announced by compromised nodes will not do harm to the timesync service if the timesync beaconing frequency is relatively low. An attack may compromise a node by announcing a high volume of timesync messages, with conforming timestamps. In this case, it is possible that our detection algorithm may mistakenly regard these incorrect timestamps as correct ones. However, as long as the compromised nodes do not form a majority within the neighborhood, such attacks can be detected by checking consistency among all neighboring nodes. The attacker may strength the attack by compromising a larger number of nodes in the neighborhood. But doing so makes the attacker being easily caught since other nodes would detect an abnormal high beaconing frequency. This creates a dilemma for the attacker that either he must risk being caught when falsifying the time too much, or he will not do any harm if staying below the detection threshold [7].

Our security mechanism builds on top of statistical detection mechanism instead of authentication mechanism. It cannot completely defend against small disruptions which do not exhibit significantly abnormal statistical patterns. For example, if an attacker reports incorrect timestamps whose deviations are still within the maximum allowable drifts, RTSP may not be able to detect such incorrect timestamps, resulting in synchronization errors in the timesync process.

There are other attack types that our security mechanism cannot work against, e.g., DoS (Denial-of-service) attack in which an attacker may jam the communication channel [16], and thus disrupt the time synchronization. The latest 802.15.4e [6] standard employs channel hopping to improve the protocol reliability. It would be interesting to devise security mechanisms with channel hopping to defend against such attacks.

E. Implementation with IEEE 802.15.4e and IETF RPL

Currently, we implement RTSP based on TinyOS CSMA MAC protocol. We discuss how RTSP can be implemented with the IEEE 802.15.4e standard and IETF RPL protocol.

IEEE 802.15.4e [6] is chartered to define a MAC amendment to the existing standard 802.15.4-2006, to better support industrial markets. Compared with TinyOS MAC, it has the following features: (1) Time is slotted and globally synchronized. (2) 802.15.4e incorporates time synchronized channel hopping (TSCH) to improve the link reliability. (3) Instead of a constant back-off, 802.15.4e uses an exponential back-off timer.

We can modify the 802.15.4e time synchronization messages to employ MAC layer timestamping. Each timesync message piggybacks a 4-byte timestamp in order to achieve accurate synchronization between the sender and the receiver. The 802.15.4e standard uses a hard-coded resynchronization period to maintain synchronization. This often results in "over-synchronization" and hence a waste of energy. We can employ an optimized approach with adaptive synchronization [1] to further reduce the energy waste. With these timesync messages, we can use the proposed algorithm to enhance the timesync process in 802.15.4e.

IPv6 Routing Protocol for Low Power and Lossy Networks (RPL) [18] is a routing protocol specifically designed for Low power and Lossy Networks (LLN) compliant with the 6LoWPAN protocol. RPL is based on the topological concept of Directed Acyclic Graphs (DAGs). RPL organizes nodes as Destination-Oriented DAGs (DODAGs), where most popular destination nodes (i.e. sinks) or those providing a default route to the Internet (i.e. gateways) act as the roots of the DAGs.

To maintain the DODAG, each node periodically generates RPL control messages (e.g., DIO messages [18]) triggered by a trickle timer. The key idea of the trickle timer technique is to optimize the message transmission frequency based on network conditions. In a nutshell, the frequency is increased whenever an inconsistent network management information is received for faster recovery from a potential failure, and decreased in the opposite case. We can reuse RPL control messages to piggyback RTSP timestamp information in order to synchronize a RPL network. Such a modification is only needed when RPL is used without an underlying synchronization service. In order words, if 802.15.4e is used together with RPL, no modification is required for RPL since we can modify 802.15.4e to support secure time synchronization.

V. Evaluation

We implement and evaluate our approach on the MicaZ platform in TinyOS 2.1.1. For comparison, we also implement ATSP and SMTS. Table I shows the memory overhead of three approaches on the MicaZ platform. We also evaluate these approaches via TOSSIM simulations. Since TOSSIM only supports MicaZ, the simulation is performed for MicaZ nodes. [1]

TABLE I: Memory overhead of three approaches (kB) on MicaZ.

	ATSP	SMTS	RTSP
ROM	4.6	7.0	7.7
RAM	2.2	2.2	2.8

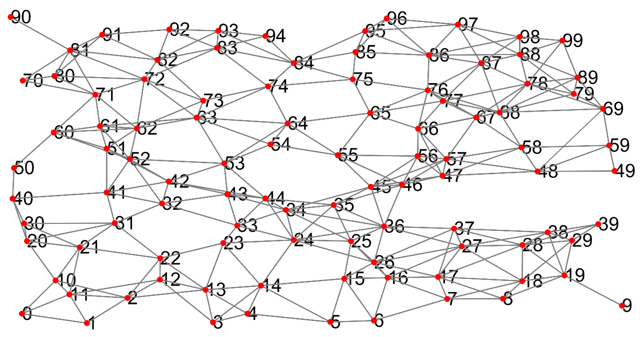

Fig. 7: Network topology

Fig. 7 shows the network topology consisting of 100 sensor nodes deployed in a $15 \times 15\text{m}^2$ area. There is an edge between two nodes if the communication is relatively reliable, i.e., the received power is larger than -80dBm.

All protocols employ distributed algorithm for achieving synchronization. The timesync beacon messages are sent every 5 seconds. Each sensor node performs one synchronization every 10 received beacon messages. Such a synchronization process is called a synchronization iteration. Since there are no reference nodes, we use the maximum time difference between two nodes as the metric for synchronization accuracy.

A. Comparative Study

Fig. 8 compares the performance of these protocols when there is no attack. We can see that all protocols eventually converge.

Fig. 9 compares the performance of these protocols under manipulation attack. Nodes 17, 37, and 53 are attackers. Each attacker broadcasts erroneous timestamp s+w where s denotes the real sending time and w is randomly chosen from [5,10]s. We can see that all protocols are robust against manipulation attacks.

We proceed to see how these protocols perform when there are Sybil attacks. Again, nodes 17, 37, and 53 are attackers. Each attacker broadcasts erroneous timestamps every 5 seconds, randomly presenting identities of neighboring nodes. We can see that the accuracy of both ATSP and SMTS degrade.

[1]We release our code at http://www.emnets.org/pub/rtsp.zip.

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/TII.2015.2495147, IEEE Transactions on Industrial Informatics

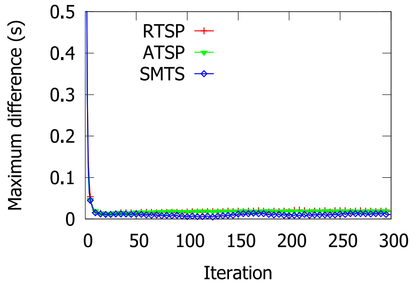

Fig. 8: No attack

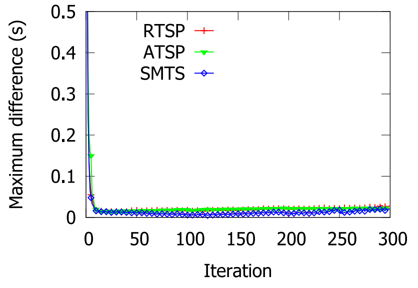

Fig. 9: Under manipulation attack

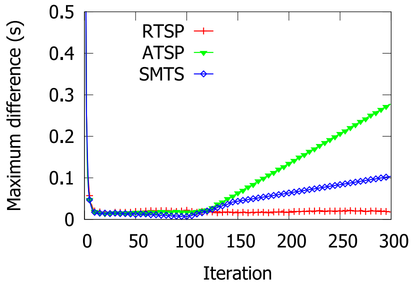

Fig. 10: Under Sybil attack

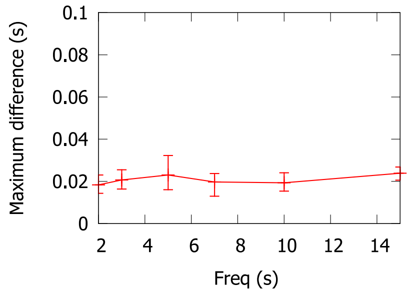

Fig. 11: Impact of attack frequency

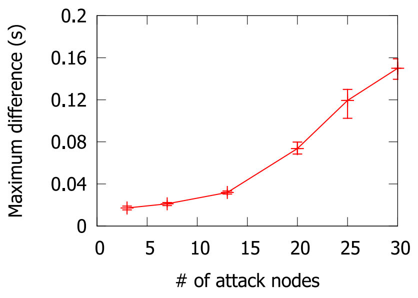

Fig. 12: Impact of # of attackers

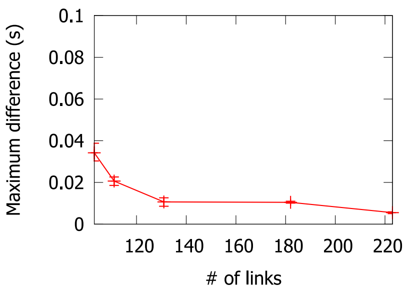

Fig. 13: Impact of network density

This is because there will be many nodes under attack. Those nodes will be considered abnormal by nearby nodes. Hence, both ATSP and SMTS lose many opportunities for synchronization. However, RTSP is robust against such attacks since it can differentiate correct timestamps from incorrect timestamps. Hence it can still exploit those correct timestamps for synchronization.

B. System Insights

We reveal more system insights of RTSP under Sybil attack by varying different parameters.

Fig. 11 shows the impact of beaconing frequency under Sybil attack with nodes 17, 37, and 53 as attackers. We can see that our protocol is not sensitive to the beaconing frequency.

Fig. 12 shows the impact of the number of attackers randomly chosen in the network topology shown in Fig. 7. We can see that the synchronization accuracy degrades when there are more attackers.

Fig. 13 shows the impact of network density with nodes 17, 37, and 53 as attackers. We vary the network density by selecting different number of links into our network topology. A small number of links represents a sparse network while a large number of links represents a dense network. We can see that our protocol is more robust when the network density is high. This is because a node has more opportunities for synchronization in a dense network.

VI. Conclusion

In this paper, we propose RTSP, a robust and secure timesync protocol employing a novel graph theoretical approach. Different from previous secure timesync protocols, RTSP is able to perform anomaly detection at per-message level, instead of node level. This fine-grained detection ability enables RTSP to become robust against Sybil attacks as well as manipulation attacks. Extensive experimental results show the effectiveness of our proposed protocol.

There are multiple future research directions. First, we would like to devise mechanisms against Sybil attacks for existing centralized protocols. Second, it would be interesting to devise security mechanisms with channel hopping to defend against jamming attacks and DoS attacks.

Acknowledgment

This work is supported by the National Science Foundation of China under Grant No. 61472360, Zhejiang Provincial Platform of IoT Technology under Grant No. 2013E60005, and Zhejiang Commonwealth Project under Grant No. 2015C33077.

References

[1] D. Stanislowski, X. Vilajosana, Q. Wang, T. Watteyne, and K. Pister, "Adaptive synchronization in IEEE802.15.4e networks," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 795–802, 2014.
[2] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.
[3] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The Flooding Time Synchronization Protocol," in *Proc. of ACM SenSys*, 2004.
[4] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. of ACM SenSys*, 2003.
[5] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. of ACM/IEEE IPSN*, 2009.
[6] IEEE Standard 802.15.4e, "Part. 15.4: Low-rate wireless personal area networks (LR-WPANs) amendment 1: MAC sublayer," 2012.
[7] X. Hu, T. Park, and K. Shin, "Attack-tolerant time-synchronization in wireless sensor networks," in *Proc. of IEEE INFOCOM*, Apr. 2008.
[8] J. He, J. Chen, P. Cheng, and X. Cao, "Secure time synchronization in wireless sensor networks: A maximum consensus-based approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1055–1065, Apr. 2014.
[9] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proc. of ACM/IEEE IPSN*. ACM, 2004, pp. 259–268.

[10] C. Lenzen, P. Sommer, and R. Wattenhofer, "PulseSync: An efficient and scalable clock synchronization protocol," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–1, 2014.
[11] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. of ACM/IEEE IPSN*, Apr. 2011, pp. 73–84.
[12] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proc. of ACM SenSys*. ACM, 2005, pp. 142–153.
[13] 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.
[14] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time Synchronization in WSNs: A Maximum-Value-Based Consensus Approach," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 660–675, 2014.
[15] S. Ganeriwal, S. Capkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc. of ACM WiSe*. ACM, 2005, pp. 97–106.
[16] K. Sun, P. Ning, and C. Wang, "TinySeRSync: Secure and resilient time synchronization in wireless sensor networks," in *Proc. of ACM CCS*. ACM, 2006, pp. 264–277.
[17] S. M. M. Rahman and K. El-Khatib, "Secure time synchronization for wireless sensor networks based on bilinear pairing functions," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, DOI: 10.1109/TPDS.2010.94, 2010.
[18] O. Gaddour and A. Koubâa, "RPL in a nutshell: A survey," *Comput. Netw.*, vol. 56, no. 14, pp. 3163–3178, 2012.
[19] H. Song, L. Xie, S. Zhu, and G. Cao, "Sensor node compromise detection: the location perspective," in *Proc. of WiCOM*. ACM, 2007, pp. 242–247.

APPENDIX

A. Proof for Theorem 1

Proof: Since $m_i \sim m_j$ and $m_j \sim m_k$, from the definition, we have:

$$1-\rho_m \leq \frac{m_j.t_S - m_i.t_S}{m_k.t_R - m_j.t_R} \leq 1+\rho_m$$

We denote $S_i = m_i.t_S$ and $R_i = m_i.t_R$ to simplify the notations. We have:

$$(1-\rho_m)(R_j - R_i) \leq S_j - S_i \leq (1+\rho_m)(R_j - R_i) \quad (6)$$

Similarly,

$$(1-\rho_m)(R_k - R_j) \leq S_k - S_j \leq (1+\rho_m)(R_k - R_j) \quad (7)$$

Adding Eq. (6) and Eq. (7) yields:

$$(1-\rho_m)(R_k - R_i) \leq S_k - S_i \leq (1+\rho_m)(R_k - R_i) \quad (8)$$

By the definition, $m_i \sim m_k$. Therefore, the theorem holds. ■

B. Proof for Theorem 2

Proof: We show the proof by induction. For $\overrightarrow{G}_1$, it is easy to see that the theorem holds. We assume $c[1],...,c[i-1]$ store the size of maximum clique containing v_p for the subgraph $\overrightarrow{G}_p$, $\forall p: 1 \leq p \leq i-1$. We will show that $c[i]$ also stores the size of maximum clique containing v_i for the subgraph $\overrightarrow{G}_i$. It is true when there are no edges from v_p $(1 \leq p < i)$ to v_i. When there are edges, the algorithm will select a vertex index p according to Eq. (5). First, $c[i]$ corresponds to a clique containing v_i: the existence of edge (v_p,v_i) implies the existence of all edges from lower vertices in the clique containing v_p to v_i (according to the *transitive* property). Second, $c[i]$ corresponds to the maximum clique containing v_i since the algorithm iterates all lower vertices and select the one that maximizes $c[i]$. Therefore, $c[i]$ stores the size of maximum clique containing v_i for $\overrightarrow{G}_i$, and we can find the maximum clique size out of all $c[i]$, $\forall i: 1 \leq i \leq n$. The theorem holds. ■

Wei Dong received his BS and Ph.D. degrees from the College of Computer Science at Zhejiang University in 2005 and 2010, respectively. He was a Postdoc Fellow at the Department of Computer Science and Engineering in Hong Kong University of Science and Technology in the year 2011. He is currently an Associate Professor in the College of Computer Science in Zhejiang University. His research interests include Internet of Things and sensor networks, network measurement, wireless and mobile computing. He is a member of IEEE and ACM, and a senior member of CCF (China Computer Federation).

Xiaojin Liu received her BE degree in Computer Science from Xidian University in 2013. She is currently a graduate student in the College of Computer Science in Zhejiang University. Her research interests include network measurement and wireless sensor networks.