Secured Flooding Time Synchronization Protocol

Ding-Jie Huang, Kai-Jie You, and Wei-Chung Teng
Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology,
Taipei, Taiwan
Email: {D9515002, M9715031, weichung}@mail.ntust.edu.tw

***Abstract*—This research aims to complement security vulnerability of Flooding Time Synchronization Protocol (FTSP), which is currently one of the most sophisticated approaches for time synchronization in wireless sensor networks (WSNs). FTSP has advanced features like implicitly dynamic topology and high accuracy time, but its original design does not consider security issues. In order to defend against attacks from malicious nodes, we propose several techniques to reinforce the structure of FTSP. A reference node selecting mechanism is proposed to reduce the effect of multiple reference nodes, and four filters are proposed to defend against $seqNum$ attack, global time attack, and node replication attack. Experiments of the above attacks are performed and the results show that all these attacks can be defended when sequence number blacklist filter and global time blacklist filter are adopted. Furthermore, fluctuation filter helps to reduce the data collection time from 10 sending cycles to 5 cycles when detecting global time attack.**

I. Introduction

Since WSN based applications mature in the last decade, security issues in wireless sensor networks have become an important research field. Due to its inherent limitations on energy, computing power, and memory size, WSNs raise the threshold of developing practical yet effective data security technologies. In 2005, Ganeriwal et al. revealed the importance of security issues in time synchronization and proposed a secure time synchronization toolbox [1]. They also pointed out the general attacks that current time synchronization protocol, such as RBS [2], TPSN [3], and FTSP [4], may encounter. Among these time synchronization protocol, FTSP is a practical one with implementation written in NesC language. However, study shows that FTSP is not been designed with security in mind [5] and it could be easily compromised by malicious nodes in a WSN. Gheorghe et al. recently proposed a fault-tolerant FTSP to defend the attack launched by a malicious node sending inconsistent global time [6], yet there are still other vulnerabilities needed to be solved. In order to defend the existent attacks mentioned in [5], we develop several techniques after thoroughly analyzed the structure of FTSP. In this paper, we implement four new components into FTSP to filter out attacking packets from malicious node.

II. Attacks on FTSP and Current Countermeasures

A. Vulnerability in Flooding Time Synchronization Protocol

Manzo et al. analyzed FTSP and pointed out its vulnerabilities in 2005 [7]. They first pointed out that the root election is not secure in FTSP. By tracing the program of FTSP, Tanya et al. later indicated several weaknesses caused by unprotected synchronization packet and developed several possible attacks on FTSP [8]. The following paragraphs discuss the currently known attacks and their countermeasures.

1) Attack on Root Selection: Every sensor node is involved in the root selection mechanism of FTSP. At the beginning of time synchronization, each node uses its own ID as $rootID$ and sends out packets with this $rootID$. After receiving a packet from some neighbor node, a node would use the $rootID$ of the incoming packet as root if the $rootID$ is less than the current $rootID$. In order to solve the root selection problem, Tanya and Sastry proposed several methods [8]. First, they use distributed coin-flipping algorithm to pick a random leader. This algorithm makes the root selection fairer and malicious nodes can not seize the root in a easy way. However, if a malicious node launches the Sybil attack [9] with a lot of fake identities, the possibility of a successful attack would be nonneglegible. Second, they proposed using multiple reference nodes instead of single one to reduce the effect from the fake messages sent from malicious nodes. Combining with RANSAC algorithm, every node can filter out the outlier messages from malicious nodes. However, these methods may cause unstable linear regression because the data is from different nodes [5]. As a result, the root selection problem still remains an open issue in FTSP. We believe that one way to solve this problem is to introduce anther root selection mechanism and we have developed an approach which is isolated from the architecture of FTSP and thus does not effect the operation of the original FTSP. However, the detailed architecture of the new root selection mechanism is not covered here due to page limit and would be summarized in another paper.

2) Attack on 'sendingtime' field: Once a malicious node wants to disturb the time system of the network, it needs to fill fake global time into the $sendingTime$ field of the outgoing packets. If any node selects this malicious node as its reference node, it uses the wrong global time after synchronizing with the malicious node. Since any node can alter its $sendingTime$, malicious node can utilize this weakness to crash the time system of the network. This attack is sometimes combined with other attacks, such as $seqNum$ attack, so the corresponding defense approach is described in the next section. In this paper, we call this attack a *global time attack*.

3) Attack on 'seqNum' field: As stated above, theoretically only the root node can increase the value of $seqNum$. However, the $seqNum$ field is not protected, so any adversary can change this value and send out fake messages with higher $seqNum$. When any node receives this fake message, it would find the $seqNum$ inside this message higher than all current records and thus selects the malicious node as its reference node. The countermeasure of this $seqNum$ attack is to use $seqNum$ filter introduced in [5]. This $seqNum$ filter uses extended data structure to collect data from randomly chosen k nodes out of total n neighbors and choose the node with smallest $seqNum$ as reference node. Hence,the $seqNum$ attack is not working under the modified FTSP.

4) Attack on Time Synchronization Rate: This attack is launched by sending out packet with faster period. This attack sometimes accompanies with the $seqNum$ attack and the global time attack, because the malicious node should send time synchronization updates more frequently than the normal node so as to increase the possibility of affecting the time of its neighbor. Therefore, the attack itself is not harmful for the time system, but any malicious node can apply this attack to launch other attacks.

5) Attack on Node Identity: As stated above, the $rootID$ is not covered, so any node can imitate the identity of any other node by sending out a packet with the ID the same as $rootID$ or programing itself as the ID of the victim node. Therefore, a malicious node can easily launch replication attack in the network, and pretend to be a reference node or even the root node. Huang et al. proposed a clock skew based node identification method which can be applied to distinguish each different node in a wireless sensor network [10]. Uddin and Castelluccia also used clock skew as identifications of sensor motes[11]. However, this node identification can not be directly applied to FTSP and some work is necessary to adapt the clock skew identity technique to FTSP.

Even if there are several countermeasures against the attacks in FTSP [8] [5], there is no overall practical solution available that could defend against all the attacks stated above. Moreover, attacks on node identity of FTSP is a severe problem and is still left unsolved. This paper proposes an more complete approach to reinforce the structure of FTSP and is able to help defend the known attacks.

III. Security Enhancement for FTSP

A. Reference Node Selection Mechanism

The reference node selection mechanism in the original FTSP is achieved by comparing the value of $seqNum$. If the value of $seqNum$ from the incoming packet is larger than the current buffered value of $seqNum$, then one node will change its reference node ID to the ID in the incoming packet. However, a node may change its reference node several times in a short period of time due to the packet loss, and this may cause the time evaluation inaccurate because of unstable clock skew. For example, the reference node of Node 4 is Node 1 when its buffered $seqNum$ is 109 in Fig. 1b, but the packets with $seqNum$ 110 and 111 from Node 1 are lost. In this case,

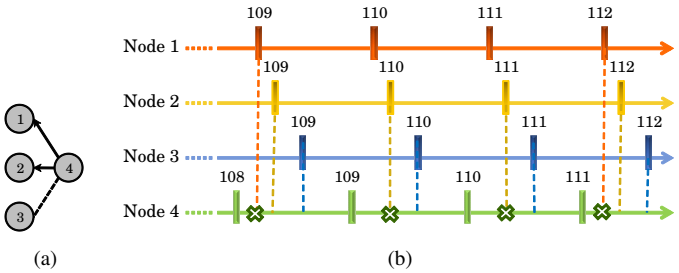

Fig. 1: The influence of packet loss on FTSP.

Node 4 changes its reference node in these two rounds and uses the synchronization information from Node 2 as shown in Fig. 1b. This behavior affects the calculation of the clock skew and makes the evaluation unstable. Sometimes the bias is tiny, our experiments show that even a bias of 1 ppm affects the evaluation of the clock skew enormously.

Hence, in order to reduce the effect of multiple reference node, unique reference node is applied to our reference node selection mechanism. If a node does not receive any synchronization packet from its reference node after N periods, it selects a new reference node from another neighbor node. A suitable value of N may depends upon the environment of the sensor node. If the packet loss rate is high, N needs to be set to a large value. In our study, N is set to 3 in the following experiments.

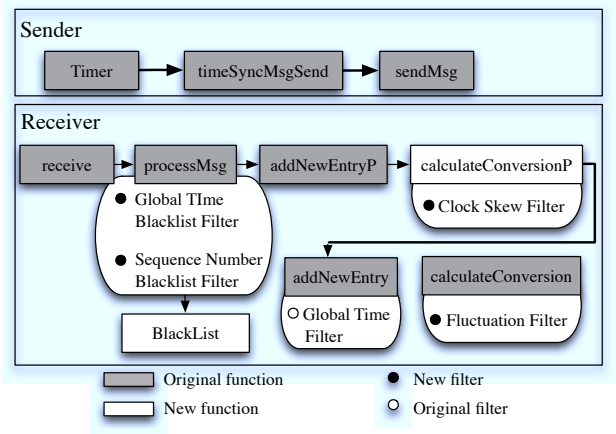

Fig. 2: New architecture with four synchronization message filters for FTSP.

B. Synchronization Message Filters

In order to enhance the security, four filters are added into the architecture of FTSP. The new architecture with four synchronization message filters is shown in Fig. 2 and we further discuss the detail of each filter in the following subsections.

1) Global Time Blacklist Filter for Neighbor Node: The global time blacklist filter for neighbor node is used to detect the difference between the current global time and the global

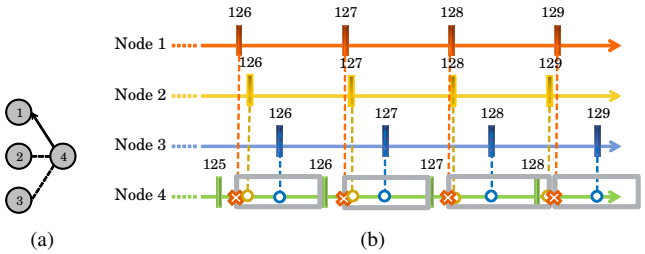

Fig. 3: Timeline example for global time blacklist filter.

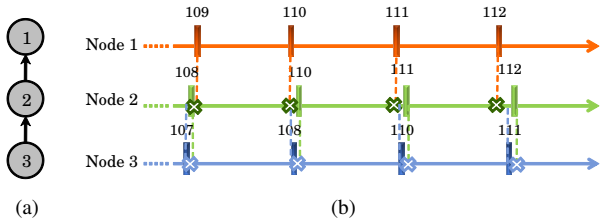

Fig. 4: The example of adjacent node with $seqNum$ difference equal to 2.

time from the incoming packet after time synchronization. If the difference is too large, the ID of the incoming packet is then added into blacklist. As mentioned above, the original global time filter in the state of *addNewEntry* is used to check the consistency between the global time of the incoming packet from the reference node and the evaluated one computed via linear regression. The subject of this filter is the reference node because any existed error needs to be filtered. Unlike the original global time filter, the subject for the global time blacklist filter is any neighbor node. The neighbor node whose global time is much different from the evaluated global time of reference node in the same period is added into the blacklist.

In Fig. 3b, the cross symbols represent the time points while receiving the packets from the reference node; the circle symbols represent the time points when receiving the packets from the other neighbor nodes. The relation between each node is in Fig. 3a. When the $seqNum$ starts from 125, Node 4 first receives the synchronization packet, which $seqNum$ is 126, from Node 1. Thus, Node 1 is the reference node of Node 4 during this period. At the same time, Node 4 will evaluate the global time by applying liner regression on the previous data from Node 1. When receiving the packets from Node 2 and Node 3, Node 4 starts to calculate the differences between the received two global time and its own global time. If at least one of the differences is too large, the corresponding node will be added into the blacklist. Furthermore, two parameters are taken into consideration while implementing the global time blacklist filter. One is the threshold L for the difference between the evaluated global time and the global time from any neighbor node; the other is the number (N) of continuous packets, which are all larger than the threshold L, from the same node. In other words, if there is a node who has N contiguous differences all larger than L, it will be added into the global time blacklist. Here, the values of N and L are depending on the synchronization message sending period. If the sending period is longer, it needs more time to synchronize to each other so the time would be much less accuracy. Then, the threshold L should be set to a larger number. On the contrary, if the sending period is shorter, the threshold L should be set to a smaller number. In our experiment, N is 3 and L is 1500 ticks.

2) Sequence Number Blacklist Filter for Neighbor Node: The purpose of sequence number ($seqNum$) blacklist filter for neighbor node is to filter out the packets with $seqNum$ significantly larger than that of the reference node. The IDs of the abnormal packets is then added into the blacklist and would not be chosen as reference node in the following synchronization. According to our practical experiment, the difference of $seqNum$ for any adjacent node should not be greater than 1 in the same period.

However, there is a special case for the $seqNum$ tiers caused by the different clock drift between each node and a node sometimes receives two synchronization packets from the same node in the same period. As the case in Fig. 4a, Node 1 is the reference node of Node 2, which is the reference node of Node 3. In this scenario, the clock of Node 1 is faster than the clock of Node 2, so Node 2 receives two packets with $seqNum$ 109 and 110 in the same period and directly changes its own $seqNum$ to 110. However, Node 3 will only receives the packets with $seqNum$ 108 and 110 from Node 2 because seqNum 109 is replaced with 110 in Node 2. Hence, this causes the difference of $seqNum$ equal to 2 on Node 3 in the same period, and the timeline example is depicted in Fig. 4b.

Thus, while implementing the sequence number blacklist filter for neighbor node, if the differences of $seqNum$ are all greater than 1 during continuous N period from the same node, the node is added into the blacklist. In this way, the condition in Fig. 4 can be prevented. Here, N is set to 3 in the following experiment.

3) Clock Skew Filter: The clock skew filter is used to detect the identity of the malicious node via applying the clock skew based identification. If an adversary wants to pretend to be the reference node or the root node, clock skew can be used to detect the identity of the malicious packets. Since the fluctuation of clock skew raises with the increase of hop numbers, we only apply the single-hop skew calculation for the reference node in the following experiment. According to the previous literature, Huang et al. mentioned that clock skew can be used as node identification in 2008 [10]. They pointed out that the fluctuations of clock skew between any two nodes are no more than 1ppm. As a result, 1ppm is applied as the threshold for the skew filter to detect the packet from the malicious node. On the implementation of clock skew filter, after receiving the synchronization packets from the reference node, each node calculates the skew between itself and the reference node. Later, if the computed skew difference

between the incoming packet and the original skew is greater than 1ppm, this message would be filtered.

4) Fluctuation Filter: The fluctuation filter is used to reduce the effect caused by the adjustment error or imitation attack lunched by malicious node. In FTSP, 4 continuos data is needed to calculate the skew between itself and global time; however, the buffer for calculating the skew would not be clear until 3 consecutive errors, between the global time from the incoming packet and the evaluated global time, are all greater than a threshold. After clearing the buffer, the node will recollect time information and calculate new clock skew. Hence, the linear evaluated value is affected significantly by the instantaneous noise for at less 3 periods.

Since the original global time can filter out the error greater than 500 ticks, it sometimes can not filter out the value within 500 ticks and the adjustment error may occur in this period. Therefore, the fluctuation filter is needed to detect the abnormal time information from the reference node. In addition, from the experimental result, the skew would not exceed the range from $1 * 10^{-4}$ to $-1 * 10^{-4}$.

As mentioned before, it needs 4 periods to collect data for calculating evaluated skew, and 3 periods to detect the error messages. Furthermore, it also needs 3 periods to collect new data to generate a new correct skew. Hence, it totally takes 10 periods to detect the errors and to recover the skew in the original design of FTSP. On the other hand, it only needs 1 periods to collect data and 1 period to detect the error with our fluctuation filter. Because while receiving the the error data in the second period, the fluctuation filter can detect the skew error and immediately clear the buffer, it can save 5 periods for detecting the skew error comparing with the original design. Thus, by applying the new fluctuation filter, it only takes 5 periods to detect errors and to recover the skew.

As a result, when receiving the time information, which causes the skew fluctuation over the range $\pm 1 * 10^{-4}$, from the reference node, the node will clear the buffer and recollect the data in order to calculate a new clock skew, because the incoming packet may come from a malicious node or under noise interference.

5) Combination Effect of Message Filters: As stated above, since any packet with the difference of the $seqNum$ greater than 1 will be added into the blacklist, a malicious node can still send out packet with the difference equal to 1. In this case, the sequence number blacklist filter can not detect this malicious packet. However, the global time blacklist filter can further detect the malicious packet and reduce the effect of this attack. In addition, combining global time blacklist filter with clock skew filter, if the skew of the global time, sent from the malicious node, is significantly different from the evaluated skew, the packet would be filtered. Thus, by utilizing proposed filters simultaneously, it can reduce the effect of the attack. Even if the packet pass through the filters, the fake global time is similar to the evaluated global time, so this attack can not affect the time system with the enhanced FTSP.

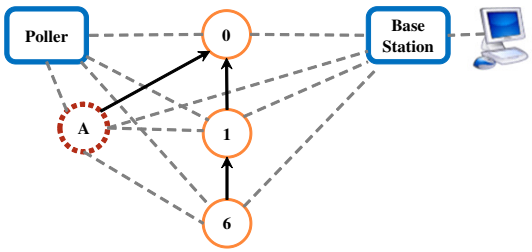

Fig. 5: Architecture of general experiment setup.

IV. Experiments

A. Exp1: Effect of Multiple Reference Nodes

The goal of this experiment is to analyze the effect of evaluated clock skew from multiple reference nodes. In this experiment, the adjacent nodes of Node 6 are Node 1 and Node 2, as seen in Fig. 6. In order to create an environment with significant interference, the global time, from the sending packet of Node 2, is subtracted with 30 ticks. After receiving the packets from Node 1 and Node 2, Node 6 collects these time messages and calculates the clock skew from these time information.

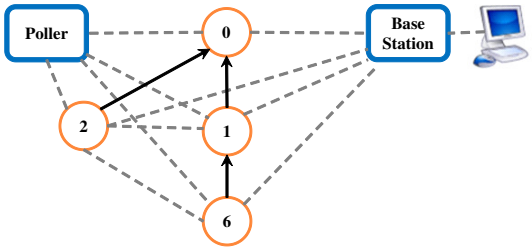

Fig. 6: Environment setup of multiple reference nodes (Exp1).

The analysis result is as in Fig. 7. The blue line is the skew variation by applying the original FTSP. Because the effect of multiple reference nodes, Node 1 and Node 2, the evaluated skew varies enormously caused by calculating different sources of global time. On the other hand, the red line is the skew variation by applying our modified FTSP. Because reference node is locked to Node 1 according to our reference node selection mechanism, Node 6 only collects the data from Node 1 and is not affected by Node 2. Hence, the evaluated skew variation is relative stable.

B. Exp2: Wrong global time sent from a malicious node

The effect of wrong global time sent from any malicious node is analyzed in the Exp2. Node A is the malicious node, who alters global time by dividing the value of original global time by 2, and sends out this fake message to its adjacent nodes. The experiment setup is in Fig. 5. Since the reference selection mechanism and global time filter are not considered in the original FTSP, it is easy to launch attacks to affect time system. As Fig. 8a indicates, Node 1 is the reference node of Node 6 before the $seqNum$ 50 so the global time in Node 6 is growing steadily. However, the malicious node appears and

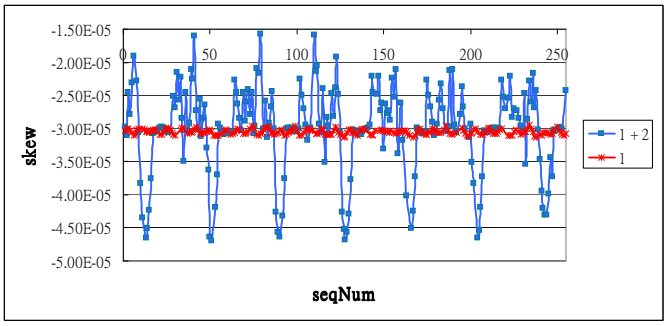

Fig. 7: Environmental result for the effect of multiple reference nodes.

launches the global time attack from $seqNum$ 50 to 100, and it causes the global time to change between the original one and the fake one in Node 6. Node 1 disappears from $seqNum$ 100 to 150, so the reference node of Node 6 is directly change to the malicious node. Thus, the malicious node successfully compromises the time system of Node 6 in this experiment.

On the other hand, the result of modified FTSP is as in Fig. 8b. Because reference node is locked to only one node with new reference node selection mechanism, the global time of Node 6 is not affected from $seqNum$ 50 to 100. Furthermore, the global time from the malicious node is much greater than the original global time, so the malicious node is added into the blacklist of Node 6. Finally, even when Node 1 disappears from $seqNum$ 100 to 150, Node 6 first checks the blacklist and does not use the malicious node as its reference node. Instead, Node 6 use the previous skew to evaluate the global time during this period. Therefore, our global time blacklist filer and reference node selection successfully defends the global time attack in this experiment.

C. Exp3: Fake $seqNum$ sent from a malicious node

The effect of the fake $seqNum$ attack launched by a malicious node is analyzed in the Exp3. Here, Node A is the malicious node and it sends out fake packets with the $seqNum$ equal to the original $seqNum$ plus 5 in every sending period, after the time is synchronized in this network. The node distribution is as in the Fig. 5.

There is no defense technique in the original FTSP, so all nodes in this network are affected by the fake $seqNum$ sent from the malicious Node 3 after the $seqNum$ is 50, as shown in Fig. 8c. The $seqNum$ attack makes all node to synchronize their time with the malicious node. On the other hand, after Node 3 joins this network, the adjacent nodes of Node 3 detect the malicious packets by applying the $seqNum$ blacklist filter and add Node 3 into the blacklist in the enhanced FTSP. Therefore, the adjacent nodes around Node 3 are not affected and do not send out the wrong messages to the other nodes in Fig. 8d. Besides, the $seqNum$ of the malicious node falls down to 90 after the $seqNum$ is 210. It is caused by the data overflow of the $seqNum$, which is a signed 8 bits parameter with the range from -128 to 127. Therefore, Node 3 thinks that Node 1 has a greater $seqNum$ so it uses 90 as the next $seqNum$ at the $seqNum$ 210. In this experiment, the attack, launched by the malicious node, is successfully defended by our $seqNum$ blacklist filter.

D. Exp4: Node ID imitating attack from a malicious node

The goal of this experiment is to analyze the effect of the attack launched by a malicious node who imitates the node ID of existing node. The malicious node A imitates the identity of Node 1 and sends out synchronization packets with the global time minus 100 ticks, as shown in Fig. 5.

In order to simplify the experiment for a better observation, we assign Node 3 to the malicious node, but the other adjacent nodes consider the malicious node as Node 1. In this way, we can simulate that there are two nodes with the same node ID and generate an diagram which is easy to analyze. Since the original FTSP can not detect the identity of each node, Node 6, which reference node is Node 1, is affected by the malicious node after $seqNum$ is 50 in Fig. 8e. There are skew variations over 1ppm during $seqNum$ 55 65, 80 85, and 95 100. This condition is caused by the imitating attack from the malicious node. However, while applying the clock skew filter, Node 6 can detect that the skew varies over 1ppm and filter out the incoming packet. As shown in Fig. 8f, after the malicious node, represented by Node 3, joins to the network, Node 6 can filter out the fake packets coming from the malicious node.

V. Conclusions

Assaults on FTSP are divided into two categories: the attacks on the root node and the attacks on normal nodes. In this paper, we do not defend against the former type and leave it as future work; we focus on the defense against the later type. The later type can be further subdivided into three different cases: (1) modifying the globalTime field; (2) modifying the $seqNum$ field; (3) reducing the frequency of sending packets. In order to defend against attacks on normal nodes, reference node selecting mechanism and several data filters are proposed. The proposed filters include black lists of neighbor globalTime filter, black lists of neighbor seqNum filter, clock skew filter and time fluctuation filter. The results of experiments show that we have successfully defended against the $seqNum$ attack and the global time attack by utilizing $seqNum$ blacklist filter and global time blacklist filter. By applying the characteristics of fixed clock skew fluctuation, around 1ppm, reference node can be successfully identified. Furthermore, the data collection time for detecting global time attack drops from 10 sending cycles to 5 sending cycles by applying the fluctuation filter. From our experimental results, the proposed filters can improve the security of FTSP and fix the vulnerabilities caused by original design.

Acknowledgment

This work was supported under the National Science Council Grants 98-2221-E-011-066-MY3.

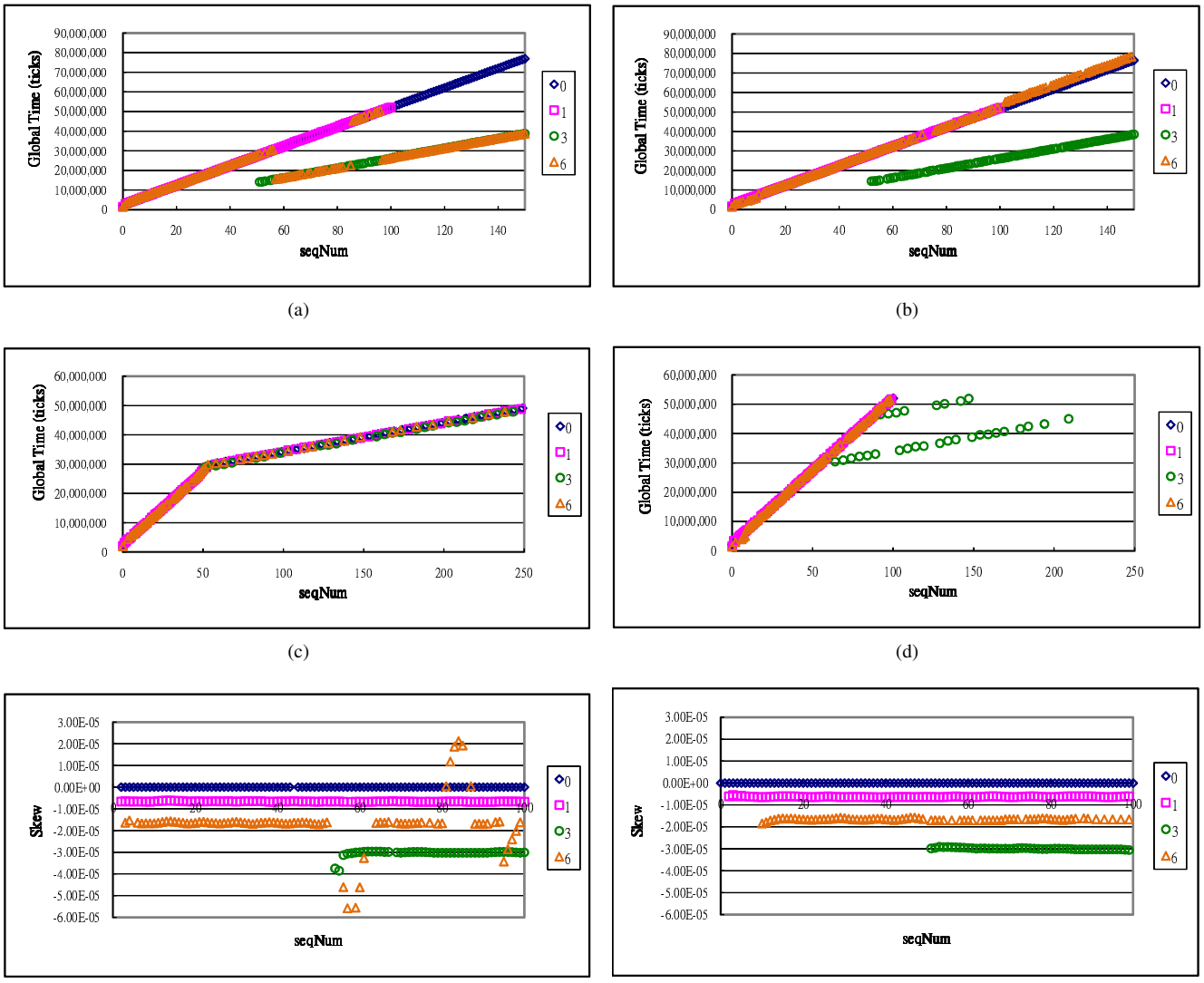

Fig. 8: (a)(b) Analysis for global time attack and corresponding defense. (c)(d) Analysis for $seqNum$ attack and corresponding defense. (e)(f) Analysis for node ID imitating attack and corresponding defense.

REFERENCES

[1] S. Ganeriwal, S. Capkun, S. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *the ACM Workshop on Wireless Security (WiSe)*, October 2005.

[2] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *OSDI '02: Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, vol. 36, 2002, pp. 147–163.

[3] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *SenSys '03: Proceedings of the First International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, USA, September 2003, pp. 138–149.

[4] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, 2004, pp. 39–49.

[5] T. Roosta, W.-C. Liao, W.-C. Teng, and S. Sastry, "Testbed implementation of a secure flooding time synchronization protocol," in *Wireless Communications and Networking Conference*, March 2008, pp. 3157–3162.

[6] L. Gheorghe, R. Rughinis, and N. Tapus, "Fault-tolerant flooding time synchronization protocol for wireless sensor networks," in *Sixth International Conference on Networking and Services (ICNS)*, March 2010, pp. 143–149.

[7] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization attacks in sensor networks," in *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, New York, NY, USA, 2005, pp. 107–116.

[8] T. Roosta and S. Sastry, "Securing flooding time synchronization protocol in sensor networks," in *First International Workshop on Embedded Systems Security*, 2006.

[9] J. R. Douceur, "The sybil attack," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK, 2002, pp. 251–260.

[10] D.-J. Huang, W.-C. Teng, C.-Y. Wang, H.-Y. Huang, and J. M. Hellerstein, "Clock skew based node identification in wireless sensor networks," in *IEEE Global Communications Conference*, 2008, pp. 1877–1881.

[11] M. Uddin and C. Castelluccia, "Toward clock skew based wireless sensor node services," in *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, March 2010, pp. 1–9.