Recovery based Time Synchronization for Wireless Networks

Anita Kanavalli, P Deepa Shenoy, Venugopal K R and L M Patnaik

Department of Computer Science and Engineering
University Visvesvaraya College of Engineering, Bangalore, India
anita.kanavalli@gmail.com

Abstract

Time synchronization schemes in Wireless Sensor Net- works have been subjected to various security threats and attacks. In this paper we throw light on some of these at- tacks. Nevertheless we are more concerned with the pulse delay attack which cannot be countered using any of the cryptographic techniques. We propose an algorithm called Resync algorithm which not only detects the delay attack but also aims to rectify the compromised node and intro- duce it back in the network for the synchronization process. In-depth analysis has been done in terms of the rate of suc- cess achieved in detecting multiple outliers i.e. nodes under attack and the level of accuracy obtained in the offset values after running the Resync algorithm.

Keywords: Time synchronization, Attacks, Security, Re- covery, Clock Adversary, Compromise nodes.

1. Introduction

Sensor Networks are made up of small devices with sensing and processing facilities equipped with a low-power radio interface. The emergence of industrial control appli- cations as compared to previous applications which were dedicated towards environmental monitoring and surveil- lance tasks. Industrial control applications demand much more with respect to security especially when monitoring of critical equipment is needed.

Time synchronization is imperative for many applica- tions in sensor networks at many layers of its design. The various examples of sensor networks include TDMA radio scheduling, reducing redundant messages by duplicate de- tection of the same event by different sensors, performing mobile object tracking, using the different sensor nodes to perform ordered logging of events during system debugging, to name a few. The effect of inaccurate time syn- chronization would be detrimental to all the above applica- tions if somehow the underlying time synchronization pro- tocol was modified by an adversary. This will affect the estimated trajectory of the object, which would greatly dif- fer from the actual one because the collaborative data pro-cessing and signal processing techniques will be greatly af- fected. Similarly, the importance of time synchronization of sensor networks can be seen in other control applications. There are several time synchronization protocols available such as RBS [2], TPSN [3], LTS [4], FTSP [5], GCS [6] that can achieve network synchronization with a high accuracy rate. Most of the synchronization methods involve some form of exchange of messages between the nodes. The syn- chronization can either be performed in a two way message exchange [3](also known as sender-receiver synchroniza- tion) or in a one way message exchange [2](also known as receiver-receiver synchronization). However, these meth- ods are not defined with security in mind, even though se- curity is one of the main challenges in sensor networks.

Motivation: Performing time synchronization of sensor networks without taking malicious attacks into consider- ation can result in inaccurate data, which could be more harmful than no data at all. Thus when sensor networks are deployed in adversarial environments such as a battlefield, the time synchronization protocol in use should be well equipped to provide security against adversaries. Therefore the idea is to develop an algorithm for time synchronization protocol.

Contribution: In this paper, we address the above men- tioned security-related issue in a three fold manner. First, we identify several attacks that can be launched by an ad- versary in a sensor network with an underlying time syn- chronization protocol which is not secure. In pulse delay at- tack, the malicious adversary

deliberately delays the trans- mission of time synchronization messages of the compromised node in order to increase the time offset between a benign node and the actual time. This type of attack can- not be addressed only by using cryptographic techniques. Also, all the presently available synchronization protocols [2, 3, 4, 5, 6] are vulnerable to this type of attack. Sec- ond, we utilize a group synchronization algorithm known as Lightweight - Secure Group Synchronization (L-SGS) [7] which is an extension of secure pair-wise communication between two nodes, to showcase how delay attacks can be detected based on a pre-specified threshold delay. Thirdly we propose an extension to the L-SGS i.e. Resync Algo- rithm, where the compromised node is brought back into the synchronization process, by running the Resync algo- rithm in the compromised node. We test the accuracy of the Resync algorithm by making a comparison between the time offset values during the synchronization process before it was compromised and the time offset value of the sensor node after it has been compromised and brought back into the network by running the Resync algorithm. This is done to illustrate the efficiency of the proposed algorithm.

Organization: The rest of the paper is organized as fol- lows: Section 2 gives overview of related work. The model and the problem definition, the analysis of the algorithm is presented in section 3. The implementation details and the performance analysis is discussed in section 4. Section 5 contain the conclusion.

2. Related work

Time synchronization mechanism has been extensively studied [2, 3, 4, 5, 6] and a survey of these protocols can also be found in [8]. There are several attacks which can be launched against sensor nodes in a sensor network which are vulnerable in hostile environments. In [9] the authors have identified few possible attacks on sensor time synchro- nization: In message modification attack, the attacker may drop, modify or even forge the exchanged synchronization messages to interrupt the time synchronization process. In the attack where messages are dropped, the attacker can drop the packets, thus prolonging the time taken to com- plete of the synchronization process. This can be done on a random or an arbitrary basis, making it more difficult to be detected. In the attack where messages are forged, the attacker may forge any synchronization message and flood the network. This will cause nodes to unnecessarily con- sume power as nodes spend time in processing unwanted synchronization messages. If some nodes run out of power, coverage holes may appear. Message dropping may be pre- vented by some misbehavior detection schemes [10]. In case of reply attack, an adversary can capture messages and replay it at a later time. The old message with an old time stamp can lead to an incorrect offset and could compromise the entire synchronization process. This can be easily prevented by using sequence numbers. When a node receives a broadcast from its parent, it can check if it is an old mes- sage. If so, the message is understood to be a replay mes- sage and discarded. In this manner, by using certain crypto- graphic techniques, replay attacks can be curbed. The mas- querade attack can be understood taking RBS as the under- lying synchronization protocol. Suppose that node A sends out a reference beacon to its two neighbors, B and C, an at- tacker E, can pretend to be B and exchange wrong time in- formation with C, disrupting the time synchronization pro- cess between B and C. We can apply certain cryptographic techniques to address this attack. Providing authentication of every exchanged message will prevent an outside attacker from impersonating other nodes and disrupting the synchro- nization process, as a result ensuring that a masquerade at- tack can be foiled.

3. Model and Algorithm

3.1. Problem Formulation

Our system consists of a network of sensor nodes which are assumed to be in their power ranges. Such sensors are called neighbors of each other. The radio link present be- tween the sensor nodes is bidirectional to allow a two-way communication between the nodes. Also the nodes are as- sumed to share secret pair-wise keys which allows secure communication. The notion of the keys is not explicitly presented in the algorithm, as several key establishment schemes are available in literature such as [11] which can be integrated with the algorithm. Each sensor node is equipped with its own clock. The clock is a combination of both hard- ware and software components. The basic functionality of the clock depends on the number of oscillations of a quartz crystal running at a particular frequency. The clock syn- chronization algorithm adjusts the local clock of a sensor node. But in the algorithm which is presented, the concern is not to synchronize the local clocks to real time. Let us represent the clock of node A as L_A . Thus, at real time t, the local clock representation is L_A (t). The difference between the clocks of any two sensor nodes at any time t is called as offset error between them. There are mainly three reasons for the local clocks of sensor nodes to repre- sent different

times which results in the occurrence of an offset error. These are listed below:

(i) The nodes present in the network start at different times. This could be due to several reasons. As a result, an inherent difference is established in the nodes local times.

(ii) Each node is equipped with a quartz crystal whose os- cillations determine the local time of the clock. The quartz crystals of the nodes may run at slightly dif- ferent frequencies. Even the smallest of differences can gradually lead to clock values diverging from each other. This error in the clocks is termed as skew.

(iii) Due to environmental conditions like change in tem- perature, etc and aging over a period of time, the frequency of the sensor node clocks can tend to vary. This kind of error is known as drift.

Therefore, we can mainly recognize three types of errors which can be present in sensor node clocks. First, of **offset** $\delta(A, B)$ which is the difference between the clocks of two sensor nodes A and B. Represented as follows:

$$\delta(A, B) = L_A(t) - L_B(t)$$

Second, skew $\eta(A, B)$ which occurs due to varying fre- quencies in nodes A and B. It is the first derivative of offset with respect time real time. Represented as following:

$$\eta(A, B) = \frac{\partial L_A}{\partial t} \frac{\partial L_B}{\partial t}$$

Third, **drift** $\lambda(A, B)$ which occurs due to ambient condi- tions such as temperature between sensor nodes A and B. It is the second derivative of offset with respect to real time. Represented as message to B, it is received at time $T2 = T1 + \delta + d$. Sim- ilarly, when Node B sends the acknowledgment message back to Node A at time T3, it is received by Node A at time $T4 = T3 - \delta + d$.

$$T2 = T1 + \delta + d \quad (1)$$

$$T4 = T3 - \delta + d \quad (2)$$

In the acknowledgment message ack, Node B sends times T_2 and T_3 to Node A. Thus, node A can calculate the offset delta and end- to- end delay.

$$\delta = (T2 - T1) - (T4 - T3) \quad (3)$$

$$d = (T2 - T1) - (T3 - T4) \quad (4)$$

In the group synchronization algorithm, based on the value of d, the nodes synchronize with each other only if their delay value has not crossed a pre-calculated threshold value based on the maximum, minimum, average and the standard deviation of the delay values of the nodes in a specific run. This process has been described in the following section. Both the receiver-receiver and sender-receiver schemes of synchronization can fall prey to the pulse delay attack. The delay attack in sender-receiver synchronization is illustrated in figure 2. Here, Node A and B are involved

$$\lambda(A, B) = \frac{\partial^2 L_A}{\partial^2 t} \frac{\partial^2 L_B}{\partial^2 t}$$

In order to understand the pulse delay attack, it is neces- sary to see how synchronization is performed between a pair of nodes. This can be broadly classified as receiver-receiver and sender-receiver. The sender-receiver synchronization is illustrated in the following figure 1. As we can see from

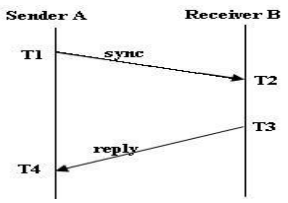

Figure 1. Sender-Receiver Synchronization

figure 1 Node A sends sync message to node B at time T1 which is received by Node B at T2. Node B then acknowl- edges the synchronization message by sending an ack reply to Node A at T3. This is received by Node A at time T4. Here T1 and T4 are times measured as per the local clock L_A of Node A and T2 and T3 are times measured as per the local clock L_B of Node B. If d is the end-to-end delay that is present between the nodes and delta is the offset between the nodes clocks, then when Node A sends synchronization

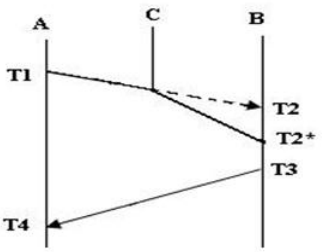

Figure 2. Pulse Delay Attack

in synchronization process as described in figure 1. But, at- tacker C (attacker and adversary are used interchangeably throughout this paper), launches a pulse delay attack, where it can jam the synchronization pulse which is sent from Node A to Node B. In the absence of the attack this mes- sage should have been received by Node B at time T2. But the adversary can capture the pulse, store it in its memory and replay it back after a delay §. Thus, the message is re- ceived by Node B at time $T2* = T2 + §$ and from equation (1), $T2* = T1 + \delta + d$ where § is the delay which has been introduced by the adversary, which causes both the clock offset and end-to-end delay to change by an additive factor of §/2, as illustrated by the following:

$$§ = (T2 - T1) - (T4 - T3) + §2$$

$$d = (T2 - T1) - (T3 - T4) + §$$

The offset and end- to- end delay calculated by Node A, results in inaccurate synchronization and Node A is referred to as compromised node. The adversary has no control over the calculation of the end-to-end delay and this observation has been used in [7] to develop the protocols to detect the compromised nodes.

3.2. Problem definition

Implementation of the modified L-SGS algorithm. The algorithm uses the broadcast property of the wireless chan- nel to broadcast messages from the central to the other nodes. The times T_{ij} are measured by the local clock of the node, where T_{ij} is the Time i, received by Node j. Each node has to keep track of four sets of time, T_i, i=1 to 4 in or- der to calculate d_j and δ_j, if not compromised. Any of the nodes in the group can serve as the central node, provided that the collisions in the wireless channel do not cause any drop of packets to and from the central node. The imple- mentation of Resync protocol in order to counter the ex- ternal attack and re-enter into the synchronization process after being compromised.

3.3. Assumptions

In this model, we assume that all the sensor nodes are aware of each others group membership. They are present in each others power ranges. Any two nodes can authenti- cate each other using pair-wise secret keys and authentica- tion schemes such as MACs (not presented as a part of the following algorithm). An explicit leader algorithm is not needed; any node which broadcasts the SYNC message becomes the central node till the synchronization process is completed.

3.4. Algorithm

The L-SGS algorithm [7] is a group synchronization al- gorithm which was developed in order to counter external attacks and message modification attacks. As prevention of message modification can be done using

several crypto- graphic techniques, we have chosen to concentrate mainly on the attack resilience component of the algorithm. The modified L-SGS algorithm is an extension of Secure Pair- wise Synchronization (SPS) [7]. This section defines the system model which is required to implement the modi- fied L-SGS. In this model, we assume that all the sensor nodes are aware of each others group membership. They are present in each others power ranges. Any two nodes can authenticate each other using pair-wise secret keys and authentication schemes such as M AC s (not presented as a part of the following algorithm). The algorithm uses the broadcast property of the wireless channel to broadcast messages from the central to the other nodes. The times T_{ij} are measured by the local clock of the node, where T_{ij} is the

Time i, received by Node j. Each node has to keep track of four sets of time, T_i, i = 1 to 4 in order to calculate d_j and δ_j, if not compromised. Any of the nodes in the group can serve as the central node, provided that the collisions in the wireless channel do not cause any drop of packets to and from the central node. An explicit leader algorithm is not needed; any node which broadcasts the SY N C message becomes the central node till the synchronization process is completed. The modified L-SGS is executed as follows:

Table I
L - SGS Algorithm

1. $N_C \rightarrow N_i$ [SY N C]∗i=1 excluding c.
2. $N_i(T_{1i}) \rightarrow N_C(T_{2i})[N_i \| N_C \| REPLY]$
3. $N_C(T_{3i}) \rightarrow N_C(T_{4i})[N_i \| N_C | T_{3i}]*$
4. Each node N_i, $i = 1 \ldots n$ calculates end-to-end delay
$$d = ((T_{2i} - T_{1i}) - (T3i - T_{4i}))/2;$$

If d <= D then offset $\delta = ((T_{2i} - T_{1i}-) - (T_{4i} - T_{3i}))/2$, else Node N_i labeled as a compromised node and runs RESY N C algorithm. In this algorithm, N_C rep- resents the central node and N_i represents the remaining nodes in the group, i.e. excluding the central node itself. Initially in Step 1, the central node sends a synchronization message to all the other nodes in the groups. This message is broadcasted by the central node instead of multiple uni- casts. Once, the central node N_C broadcasts the SY N C message to all the other nodes in the group, the group mem- bers i.e. N_i, i = 1 . . .N excluding the central node c, reply to this synchronization message by returning their ids to the central node in the REP LY message at time T_{1i}. This time is recorded by each of the nodes as it is used in the delay computation. This is done in Step 2 of the proto- col. In Step 3, the central node waits for all the other nodes to reply back to it, before it includes the time of receipt of the replies from the other nodes and the time of sending of the current message in the message. This message is then broadcasted to the other nodes of the group. Thus, it is very much important that the central node which is starting the synchronization process be such that the replies form the other members of the group reach the central node, without any loss of messages in the wireless channel. When this message is received by all the group members, with times T_{2i} and T_{3i}, with the prerecorded time of sending the first reply message T_{1i} and the time of receipt of the final reply message T_{4i}, the group members can calculate the delay us- ing equation (iii). This delay d, like in SP S is compared with the threshold delay value D. If the delay exceeds this value then the node is said to be compromised and it has to enter the resynchronization process. If the delay is less than or equal to delay threshold, then the offset can be calculated and the synchronization of the node can be performed.

3.5. Resync Protocol

This proposed algorithm is a continuation of the mod- ified L-SGS. As compared to the L-SGS, the modified LSGS, does not abort when a node gets compromised. In- stead, the node which has been identified as the compro- mised node runs the Resync protocol in order to counter the external attack and re-enter into the synchronization process after being compromised.

The main idea behind which is used in the Resync algo- rithm is that once the outliers have been detected i.e. the malicious time offsets, they need to be excluded while cal- culating the true offsets between the nodes. This can be done by calculating the mean of the offsets of the benign nodes and approximate true time offsets.

Let Γ be the time offset data set from all the nodes and χ be the time offsets from the outlier set, then the benign time offset set is defined as $\Gamma - \chi$. Also let μ is defined as the average of the set $\Gamma - \chi$. The size of set of time offsets is n and that of the time offsets of compromised nodes is k, then is calculated as follows:

$$\mu = \sum (\Gamma - \chi)i/(n-k)$$

where I ranges from 1 to $n-k$

Therefore, when a node gets compromised, it has to ensure that it gets the true offsets from the nodes which are not compromised in the network. Accordingly, it has to take the average of these time offsets received and set its ffset to the calculated mean. The following algorithm, implements the following concept.

The Resync algorithm executes as follows:

Table II Resunc Algorithm

1. $N_X \rightarrow N_i |N_X \| \text{COMPROMISED}]$*
2. $N_i \rightarrow N_X [\delta_i \| \text{OFFSET }]$
3. N_X calculates the average of the offsets of the remain- ing benign group members and adjusts its clock.
4. $N_X \rightarrow N_i [N_X \| \text{RESYNC}]$

Node N_X represents the node which has been compro- mised. This has to be informed to all the other group mem- bers taking part in the synchronization process. This is achieved in Step 1, where the compromised node includes its node id and broadcasts it to all the nodes in a COMPRO- MISED message. In Step 2, as a reply each of the benign nodes, include the offset that they calculated as a part of the modified L-SGS in the OFFSET message and transmit it to the compromised node. The purpose of doing this is that the compromised node is able to use the offsets cal- culated by each of the nodes by computing the average of these offsets and setting its clock accordingly. This is per- formed in Step 3 of the algorithm and accordingly the local clock of the compromised node is brought back to the un- compromised state. In the last step, this re-synchronized node informs the other nodes about its new status by broad- casting the RESYNC message. Once this is done, the entire synchronization cycle can be restarted.

4. Implementation and Performance Analysis

4.1. Implementation of modified LSGS

Our implementation of the modified L-SGS is simulation oriented. The simulation was performed using a recently developed simulator Castalia built specifically for Wireless Sensor Networks. Castalia is built on top of open source mobile network Omnet++. Castalia simulator is described in [13].One of these features is CPU clock drift which can be incorporated into the simulation to provide a real world working of the sensor nodes. Also several types of wire- less channel modes are available such as with interference, ideal channel, etc. These variations help in studying the synchronization process under several different variations. For our software simulation, we introduced an in-built CPU clock drift for the individual sensor nodes. We have also introduced wireless interference in the simulation in order to understand loss of packets which occurs during the run- ning of the algorithm. End-to-end-delay In order for the algorithm to function correctly, the calculation of threshold delay value D needs to be performed. This is based on the delay values which are generated by the sensor nodes when none of the nodes have been compromised. The threshold value depends on the average of the delay values d_{avg} and their standard deviation σ. The delay values in every run should fall in the interval $[d_{avg} - (n - \sigma), d_{avg} + (n * \sigma)]$ where n is a positive integer. A sample run and the calculation of the threshold value has been illustrated below:

For the above sample run, there are nine nodes taken into account, one acting as the central node. The delays have been plotted when no node is compromised. Based on the threshold calculation presented above average de- lay $d_{avg} = 0.0198446$s and $\sigma = 0.00794624$. The re- quired $n = 3$ and thus the threshold is calculated to be $D = 0.04368332$s. This value is then used to detect outliers in the subsequent runs. Based on the threshold delay calcu- lation, the presence of compromised nodes can be detected as described in the algorithm. The following figures illus- trate how the algorithm is able to pick up on the presence of a node under pulse delay attack based on the threshold value. The first scatter graph in figure 4 plots the uncom- promised nodes and their respective delays. All the delays are below the threshold which is represented by the dashed line. In the second scatter graph in figure 5, Node 5 gets compromised as it is under a delay attack. This forces its delay to go beyond the threshold value D and hence is eas- ily detected by

the L-SGS algorithm as the compromised node.

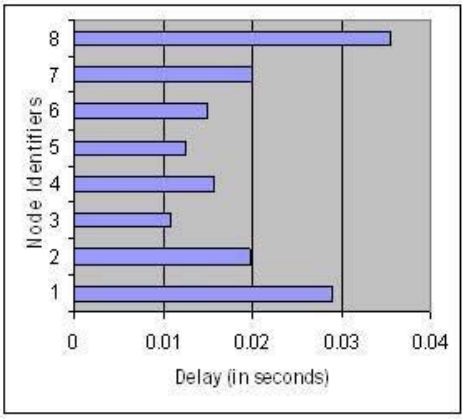

Figure 3. End-to-end delay in a sample run

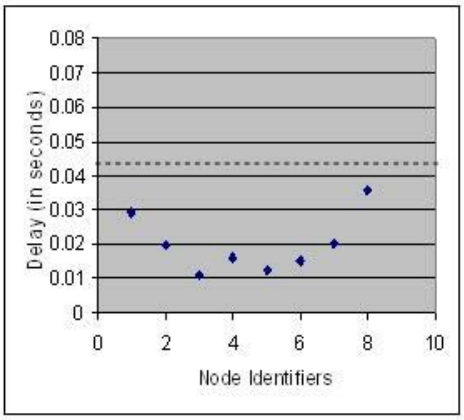

Figure 4. End-to-end delays of non compro- mised nodes

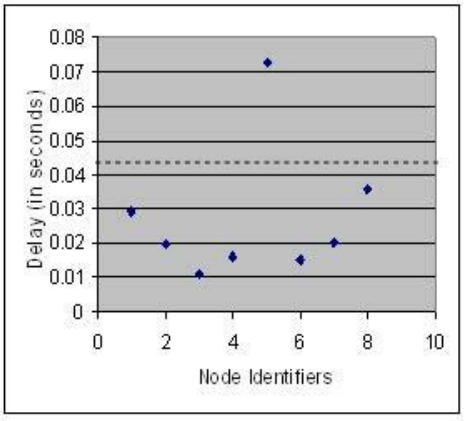

Figure 5. End-to-end delay when Node 5 is compromised

4.2. Performance Evaluation

As it has been stated in [7], there are mainly three signif- icant contributors to end-to-end delay. They are the following:

(1) Waiting time at the medium access control layer to ac- cess the channel. This delay can vary from a few mi- croseconds to a few minutes.

(2) Time taken in transmitting the packet bit - by - bit by the sender node. This time depends on the packet size and the radio speed and hence tends to be deterministic in nature.

(3) Propagation time over the wireless link between the sender and receiver node which tends to be not more than a few nanoseconds.

Keeping these factors in mind, the performance of the LSGS was evaluated on the basis of Successful Detection Rate (SDR). SDR is defined as the ratio between the total numbers of compromised nodes detected by the L-SGS and the total number of compromised nodes that are present in the network. Three different delay values of 30ms, 50ms and 100ms were introduced in the compromised nodes. Also, these delays were tested by increasing the number of compromised node from 1 to 4. The histogram consisting of this data has been presented below:

As we can see from the clustered histogram in figure 6, the successful detection rate varies from 0 to 67% based on the number of compromised nodes. When there are two nodes under attack, only three are detected giving only 50% detection rate. On the other hand, introduction of a large delay Successful-Detection-Rate as that of 100ms ensures
100% detection rate of the compromised nodes. An intro- duction of delay of 50ms gives nearly 100% accuracy, but this rate falls as the number of compromised nodes tend to increase. Thus, the algorithm works very efficiently when large delays are introduced, but its detection rate falls as the delay in the sensor nodes decreases. To achieve full accu- racy in the outlier detection is a part of out future works.

4.3. Implementation of Resync Protocol

The Resync protocol has been proposed as an extension of the modified L-SGS protocol. In the modified L-SGS, we ensure that the detection of the nodes under delay at- tack is being performed. Once a node has been labeled as a sensor node, the Resync algorithm is run in the node, to ensure that its offset is adjusted to approximately the cor- rect time. As the Resync algorithm is an extension of the L-SGS algorithm, it uses the same primitives as the L-SGS that have been described in the previous section. This in- cludes the implementation of the algorithm on the Castalia framework built on top of Omnet++. Thus, we can see that unlike the L-SGS, the Resync algorithm is not only resistant to the pulse delay attack but also aims to ensure that a com- promised node can be corrected and brought back into the synchronization process of the sensor network. This makes the senser network system efficeint.

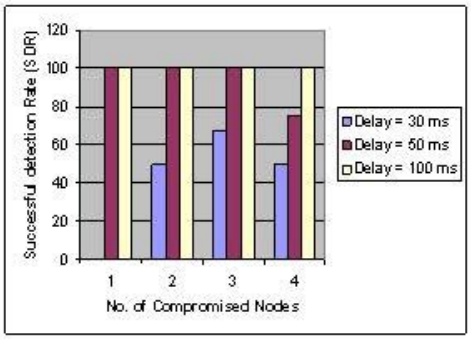

Figure 6. Successful Detection Rate for mul- tiple outliers using different delay values

4.4. Performance Evaluation

The Resync algorithm is responsible for ensuring that if a node is compromised, it broadcasts its status to the other nodes which are present in its group and using the time off- sets of the benign nodes be able to correct its offset. The- oretically, the mean of the offsets of all the benign nodes should be taken into account by the compromised node. But, in reality when packets containing the offsets are trans- mitted in the wireless channel, some of them might get lost. As a result, in the simulation for the current scenario only those offsets can be used in calculating mean which arrive at the compromised node before the simulation of the syn- chronization process ends.

As the Resync algorithm is an extension of the modified L-SGS, it is very important to see how accurate Resync al- gorithm is. This has been done in the following manner. Us- ing Castalia, we first ran the simulation

to get the offsets of the sensor nodes when none of the nodes are compromised. This is the true time offset data set of the sensor nodes. In the next run, one of the nodes is compromised as it is under the pulse delay attack. This node is detected by the modi- fied L-SGS and then corrected using the Resync algorithm. The offset which is now calculated for the compromised is compared with the original offset when the node was not compromised at all. This is repeated for each node which is present in the network, in the same run. Further, in order to see how well the Resync algorithm functions, these sets of true offset value and Resync-ed offset values are plotted against each other in order to study the algorithms accu- racy. As we can see from the figure 7, Nodes 1,3,4,5 and 7

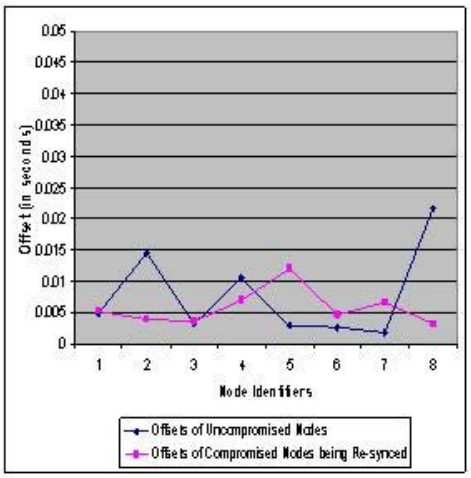

Figure 7. Run 1-Comparison of true and cal- culated offsets

i.e. five out of the eight nodes have a difference in the time offsets between 1 to 10 ms. The maximum difference that is present in the offsets is for node 8 of approximately 20ms. Similarly, in the second plot we observe that the offsets of Nodes 1,2,3,4,5 and 7 i.e. six of the eight nodes, differ with the true time offsets only by 1 to 5 ms. Here the maxi- mal difference between the time offset values is reduced to about 12ms. This trend is illustrated in figure 8, where six out of eight nodes differ in the true and determined time off- sets by 1 to 10ms. But node 5, offset is determined incor- rectly as there is large difference of approximately 20ms. We can define efficiency of the Resync algorithm as the ra- tio of the number of nodes whose difference between the true and timed offsets is between 1 to 10ms with respect to total number of runs that were tested in the sample run. Thus, as we can see from the plots above even though the Resync algorithm has an ef **f** iciency $> 70\%$ in most cases, this may still fall short in time critical areas, such as bat- tlefields. Our aim is to ensure that the Resync algorithm is able to achieve 100% accuracy rate. This is a part of our future works, where we will try to make up for the packet loss which exists in the wireless channel. By reducing the losses, we can ensure that the true time offsets are calculated much more precisely. Thus when the Resync algorithm is run, the calculated offset will also be more precise for the compromised node.

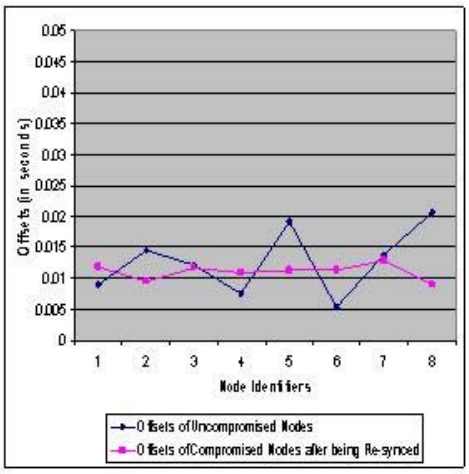

Figure 8. Run 2-Comparison of true and cal- culated offsets

5. Conclusions

In this paper, we identified various attacks that affect the standard time synchronization protocols currently available and we focused mainly on the pulse delay attack, which is not solved using cryptographic techniques. We than pre- sented the L-SGS modified to suit needs where the outliers are detected and the compromised does not abort but in- stead runs the proposed Resync algorithm in order to un- compromised the outlier. The performance of the modified L-SGS is measured using the Successful Detection Rate (SDR) where different delays are introduced into the net- work. The performance of the Resync algorithm is then measured by comparing the calculated offsets with true time offsets. Even though in most cases, the accuracy between the time offsets is about 70% or more, the proposed Resync algorithm can be used to achieve a higher accuracy rate. Thus, an efficient time synchronization sensor network sys- tem is realized.

6. FutureWorks

As a part of future works, the disparity between the off- set values needs to be reduced. L-SGS does not perform well in the presence of internal attackers i.e. when one of the nodes in the group itself acts as the adversary and com- promises the other nodes. Bringing about resilience to inter- nal attacks will in every way make L-SGS a complete time synchronization protocol, as it will be able to tolerate both internal and external attacks.

7. References

[1] W. Zhang, G. Cao, Optimising tree reconfiguration for mobile target tracking in sensor networks, in: IEEE INFOCOM, 2004.
[2] J. Elson, L. Girod, D. Estrin, Fine-grained network synchronization using reference broadcasts. In Pro- ceedings of the Fifth ymposium on Operating Systems Design and Implementation(OSDI), Boston, MA, De- cember 2002.
[3] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timesync protocol for sensor networks. In Proceedings of the First ACM Conference on Embedded Networked Sen- sor Systems (SenSys), Los Angeles, CA, November
[4] 2003.
[5] J. V. Greunen, J. Rabaey, Lightweight time synchro- nization for sensor networks. In Proceedings of the Second ACM InternationalWorkshop onWireless Sen- sor Networks and Applications (WSNA), San Diego, CA, 2003.
[6] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, The flood- ing time synchronization protocol. In Proceedings of the Second ACM Conference on Embedded Net- worked Sensor Systems (SenSys), November 2004.
[7] Q. Li, D. Rus, Global clock synchronization in sensor networks, in: IEEE INFOCOM, 2004.
[8] S. Ganeriwal, S. Capkun, C. Han, and M. Srivastava.
[9] Secure time synchronization service for sensor net- works. In WiSE, September 2005.
[10] B. Sundararaman, U. Buy, A. D. Kshemkalyani, Clock synchronization for wireless sensor networks: a sur- vey, Ad Hoc Networks 3 (3), 2005.
[11] H. Song, S. Zhu, G. Cao, Attack-resilient time syn- chronization for wireless sensor networks, 2006.
[12] S. Marti, T. J. Giuli, K. lai, M. Baker, Mitigating rout- ing misbehaviour in mobile ad hoc networks, in: ACM MOBICOM, 2000.
[13] H. Chan, D. Song, A. Perrig, Random key predistribu- tion scheme for sensor networks. In Proceedings of the IEEE Symposium on Security and Privacy, 2003.
[14] J. H. Hoepman. A. larsson, E. M. Schiller, P. Tsigas, Secure and Self-Stabilizing Clock Synchronization in Sensor Networks.