Secure topology discovery through network-wide clock synchronization

Roberto Solis Robles Jason J. Haas Jerry T. Chiang Yih-Chun Hu P. R. Kumar
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL, 61801, USA
{robsolis, jjhaas2, chiang2, yihchun, prkumar}@illinois.edu

***Abstract*—Clock synchronization is a fundamental service for many applications in wireless sensor networks. Because of its critical importance, several protocols have been proposed to achieve clock synchronization. However, most of them are not designed to work in adversarial environments, where security attacks occur. In particular, a man-in-the-middle attack, where the attacker tampers with the properties of a link, could prevent the proper operation of the clock synchronization protocol, therefore affecting every application that relies on the clock synchronization service. We present a secure network-wide clock synchronization protocol which also allows the nodes to securely discover the network topology by detecting and isolating links that behave inconsistently. This protocol is based on detecting attacks using timing information alone under certain conditions. We have developed an implementation of this protocol where we have shown that any inconsistent link found in the network is effectively eliminated.**

I. Introduction

Wireless sensor networks have been extensively used in recent years with a wide variety of novel applications that range from object tracking, to environmental and health monitoring. In most of these applications, there is a need to determine the times at which events occur, making clock synchronization a critical service. Although many clock synchronization protocols have been proposed [1]–[6], they are not designed to work in an adversarial environment. When a wireless network is deployed in an adversarial environment, the nodes become susceptible to a *wormhole attack* [7], where attackers create a link that does not behave like the other links in the network. A *man-in-the-middle* attacker tampers with the properties of a link (e.g., latency, loss rate, bandwidth), and by delaying packets during forwarding, an attacker makes legitimate nodes believe they are synchronized, when in reality they are not, since the clock synchronization protocol has no means to detect the attack.

In a previous work [8], by using an algorithm presented in [9] and [10], we proposed a protocol that is able to detect all half-duplex and some full duplex man-in-the-middle attackers, *using timing information alone*, in a single-link scenario. The work assumes that all clocks in the network are affine and it basically shows how to limit the man-in-the-middle attacker's behavior so that it can only introduce a constant delay on the link it controls.

This material is based upon work partially supported by USARO under Contract Nos. W-911-NF-0710287 and W911NF-08-1-0238, AFOSR under Contract FA9550-09-0121, and NSF under Contracts No. CNS-07-21992.

In this paper, we extend our previous work from a secure single-link clock synchronization protocol to a secure *network-wide* clock synchronization protocol, which has the following properties:

1) **Detection.** The protocol is able to detect misbehaving links and tag them.
2) **Dissemination.** Once a misbehavior is detected by a node, it can disseminate an "alert" throughout the network.
3) **Isolation.** When a node A wants to communicate with a node B and misbehaving links have been detected along one or more paths between them, node A can execute a verification process to determine which path(s) contain these misbehaving links and remove them from its internal table, effectively isolating them.

The network-wide synchronization is achieved by having each node in the network examine every path that could be used by half- and full-duplex attackers. Whenever an *inconsistent* link (i.e., a link under the control of a man-in-the-middle attacker that cannot induce a constant delay) is found, it is removed. Since we require the testing of every path in the network for attackers, our protocol allows the nodes to *securely discover the topology* of the network.

One important advantage of our protocol is that by having a secure knowledge of the network topology, it can provide further capabilities such as secure link state routing and secure source routing.

The protocol has been implemented on an Imote2 [11] sensor network testbed and it has been shown that any inconsistent links found in the network are effectively eliminated.

The organization of this paper is as follows. Section II presents background material on wormhole attacks and clock synchronization. Section III briefly describes how we securely achieve clock synchronization in a single link, and Section IV presents our secure network-wide synchronization protocol. We briefly describe our implementation and present the main results of our experimental evaluation in Section V.

978-1-4244-7138-6/10/$26.00 ©2010 IEEE

II. Background

We now define some important terms that will be used in the remainder of the paper, and briefly discuss wormhole attacks as well as related work on clock synchronization.

A. Wormhole Attack

A wormhole attacker seeks to alter message delay, that is, the time from sending to receiving, and not to alter the integrity of the messages themselves. For us, a *consistent* packet is a packet that has been delayed by a constant amount by an attacker, and such an attacker is called a *consistent* attacker. In case the attacker cannot or does not inject a consistent delay, then the attacker is called an *inconsistent* attacker. Whether the wormhole attackers are consistent or inconsistent, they can be classified according to their communication capabilities as follows:

1) A *half-duplex* attacker can either receive or transmit one message at any instant in time, but not receive and transmit at the same time.
2) A *full-duplex* attacker can receive and transmit one message at the same time.
3) A *double-full-duplex* attacker can receive two messages and transmit two messages at the same time.

B. Clock Synchronization

In [9] a trusted environment is assumed for the clock synchronization protocol, providing fundamental limits of timestamp-based clock synchronization. It is shown that location information alone is insufficient for determining communication delays over a link and node clock offsets for timestamp-based clock synchronization. This work is extended in [10] to networks of nodes using affine clocks, showing the feasibility of exactly determining both round-trip delay and clock skew relative to a global reference clock. It is proved in Lemma 3 of [10] that a transmitting node after clock synchronization can perfectly predict the time when a receiving node will receive a transmitted packet according to the receiving node's clock.

Regarding time synchronization algorithms for wireless sensor networks, several protocols have been proposed. In RBS, described in [3], an intermediate node transmits a reference packet and all nodes in the neighborhood record the time at which they receive it; they then exchange this recorded time to find their clock's difference. In TPSN [4], time synchronization is achieved in two steps; the first step is to create a hierarchical topology in the network where every node is assigned a level in the hierarchical structure, and in the second step every node at level i of the topology synchronizes to level $i-1$ by means of a two-way message exchange. In FTSP [5], time synchronization is achieved through broadcasts of messages that are timestamped at the sending and reception events, and once a number of these readings are available the clock offset and skew are calculated using linear regression. A completely asynchronous and distributed algorithm that exploits the large number of global constraints that need to be satisfied by a common notion of time in a multi-hop network is presented in [6]. This algorithm was also implemented on Berkeley motes and it was shown that it provides improved accuracy over FTSP. All of the protocols mentioned, with the exception of RBS, use MAC layer timestamping capabilities to eliminate several sources of error in the time synchronization. In [12] a time-diffusion synchronization protocol is proposed to synchronize an entire network. In [13] three methods are proposed to achieve global clock synchronization: one all-node based, one cluster based, and one based on diffusion.

As mentioned in some studies [14], [15], none of these algorithms was built with security considerations, making them vulnerable to attacks, and very few clock synchronization protocols have been proposed to handle such attacks. In [16], the authors define the *pulse-delay attack*, and propose secure protocols to handle such an attack, which uses an authentication mechanism that is only suitable for sensors with low data rates. In [17], two protocols are proposed to securely synchronize the clocks of sensor network nodes to global time servers, in a fault-tolerant manner. However, there is no mention of how the process of neighbor discovery is performed, or the possible consequences of an attack on such process.

III. Single-Link Secure Clock Synchronization

We now provide a brief overview of our previous work [8], where we extended the clock synchronization protocol of [9] and [10], to provide security against inconsistent wormhole attackers, as defined in Section II-A. The notation used in the sequel is summarized in Table I.

TABLE I
Notation

Symbol	Definition (units)
τ_A	Turn-around time of node A (s)
r	Round-trip time (s)
r'	r + delay induced by attacker (s)
m_{AB_i}	Attacker delay of packet i from A to B (s)
o_A	Clock offset of node A with respect to the global reference (s)
S_A	Clock skew of node A with respect to the global reference
S_{AB}	Relative clock skew of node B to node A, $S_{AB} = S_B/S_A$
δ_{AB}	One-way delay from A to B
N_{AB}	The total number of packets sent so far from A to B
$\rho_{AB}(n)$	B's local clock time when the n^{th} packet from A was received
$\sigma_{AB}(n)$	A's local clock time when it sent its n^{th} packet to B
$P_{AB}(n)$	n^{th} packet sent from A to B

A. Assumptions

- Clocks are affine. An affine clock is one where the local time at node **A** has the value

$$A(t) = S_A t + o_A\,,$$

where
t is the time at some master clock,
$S_A > 0$, denotes the *clock skew*, which is the relative speed of the clock at node **A** with respect to the master clock, and
o_A denotes the *offset* of node **A** with respect to the master clock.

- Nodes can accurately timestamp packets.
- When two nodes can communicate directly we say that they share a link (i.e., that they are neighboring nodes), and both nodes trust each other.
- Both endpoints of a link are half-duplex nodes, and τ_A is the time to switch from transmit to receive, or vice-versa.
- When nodes share a link, they also share a private key, which is used to provide authentication and confidentiality. By using a private key we can eliminate traditional, cryptographic man-in-the-middle attacks. There are several techniques to derive the private key [18].
- An attacker is unable to break the cryptography used in the protocol.
- The product of the relative skews between two neighboring nodes **A** and **B** must be equal to 1, that is, $S_{AB}S_{BA} \approx 1$.

B. Basic clock synchronization protocol

The basic clock synchronization between two nodes **A** and **B**, is achieved by exchanging 4 packets as shown in Figure 1, where $P_{AB}(n)$ denotes the n^{th} packet sent from node **A** to **B**. It includes the time at which it is sent, $\sigma_{AB}(n)$, and the last time at which a packet was received in the reverse path, $\rho_{AB}(n-1)$. Once the aforementioned packets have been exchanged, node **A** has enough information to estimate the relative skew of **B**'s clock with respect to **A**, $\hat{S}_{AB}$, and the one-way delay from **A** to **B**, $\hat{\delta}_{AB}$, using the following equations:

$$\hat{S}_{AB} = \frac{\rho_{AB}(n) - \rho_{AB}(n-1)}{\sigma_{AB}(n) - \sigma_{AB}(n-1)}, \qquad \hat{\delta}_{AB} = \rho_{AB}(n) - \hat{S}_{AB}\sigma_{AB}(n).$$

Once another packet is sent from **A** to **B**, $P_{AB}(n+2)$, node **B** is able to perform a similar calculation for the reverse link. Once this basic clock synchronization is performed, as [10] showed, legitimate nodes can exactly predict the time at which the packet $P(n+1)$ will be received according to the receiving node's clock, by using the following equation:

$$\hat{\rho}_{AB}(n+1) = \hat{S}_{AB}\sigma_{AB}(n+1) + \hat{\delta}_{AB}.$$

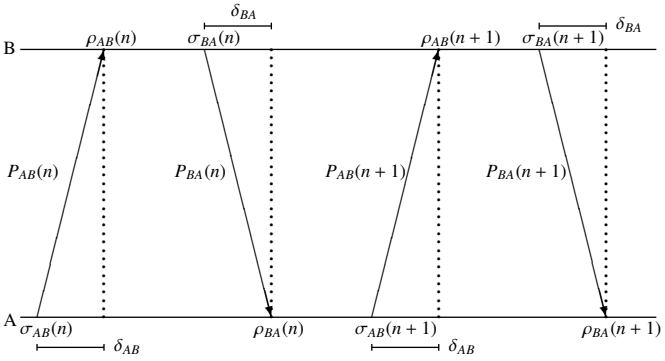


Fig. 1. Basic clock synchronization

By using this basic clock synchronization protocol, we prove in the following theorem that legitimate nodes are able to impose restrictions on the kind of additional relaying delays a wormhole attacker can impose on the packets exchanged between them.

Theorem 3.1: If two nodes having affine clocks use a fixed network where the delay between the nodes is constant, and are able to authenticate packets sent by each other, then any attacker that delays packets by an amount of time that is not constant can be eventually detected using timing information alone.

Besides showing that a wormhole attacker must be consistent in order to avoid being detected, we also showed that sufficiently distant full-duplex and half-duplex wormhole attackers will eventually be detected regardless of their consistency.

IV. Secure Network-wide Clock Synchronization Protocol

We have extended our previous work [8] from a single-link scenario to a multi-hop network where links on which wormhole attackers are detected are ignored in routing decisions.

Our secure network-wide clock synchronization protocol, which continues using the notation and assumptions stated in Section III, seeks to provide a network-wide compliant clock. That is, for any two nodes **A** and **B** in the network, the skews of **A** measured by B through k different paths between them should be approximately equal. In case the skews obtained do not obey this property, then we say the skews are *non-compliant*, which implies that one or more links have been compromised, and we remove the incorrect ones. The protocol is divided into four steps: single link check, neighborhood check, network compliance check, and removal of network non-compliance.

a) Single Link Check: In this step, every node performs clock synchronization with all of its neighbors. In other words, if there exists a link between **A** and **B**, then **A** and **B** perform a secure clock synchronization as described in Section III during this step. At the end of this step, all pairs of nodes **A** and **B** that share a link can measure the estimated relative skews $\hat{S}_{AB}, \hat{S}_{BA}$. All nodes **A** and **B** then verify that $\hat{S}_{AB}\hat{S}_{BA} \approx 1$. Each node also uses the measured skews to constantly monitor the consistency of its links.

b) Neighborhood Check: In the next step, node **A** uses the information collected in the first step and isolates misbehaving links by compiling a neighbor list, which excludes any neighbor **M** where $\hat{S}_{AM}\hat{S}_{MA} \not\approx 1$ (i.e., only valid neighbors are included, and links controlled by inconsistent attackers are excluded), and distributing this valid neighbor list to each valid neighbor. This list includes the ID's, measured skews, and measured round trip times of the neighbors reachable in a single hop. By not including the inconsistent links, a node is assured that all links incident on itself are either direct links or are controlled by consistent attackers.

c) Network Compliance Check: In this step, nodes discover the existence of nodes not directly connected to them, and check the paths that have been learned for possible misbehaving links. This is achieved by constructing a list of reachable nodes with the information received in the previous step. The construction of this list involves adding any new nodes from loop-free paths, calculating aggregate skews and

round trip times, and verifying the compliance of the end-to-end relative skews (i.e., the product of all relative skews along a path). This last verification is performed whenever a new path to a previously registered node is found. For instance, if node **A** previously found a path P_1 to node **B** through node **X** with an aggregate skew $\hat{S}_{AB,P_1} = \hat{S}_{AX}\hat{S}_{XB}$, and now finds another path P_2 to node **B** through node **Y** with an aggregate skew $\hat{S}_{AB,P_2} = \hat{S}_{AY}\hat{S}_{YB}$, then $\hat{S}_{AB,P_1}$ and $\hat{S}_{AB,P_2}$ are considered compliant if $\hat{S}_{AB,P_1} \approx \hat{S}_{AB,P_2}$ to within an acceptable tolerance. If during the construction of the list of reachable nodes, new nodes are added, meaning nodes one hop farther have been discovered, then the node constructing the list floods the updated list to its neighbors. If every node sends its list of reachable nodes to its neighbors every time there is a change in it, eventually every node will discover the network topology. We note that in this step we are actually carrying out a procedure similar to the one performed by the distributed Bellman-Ford algorithm [19], only that instead of computing a minimum cost path, we are aggregating the skew and round-trip time.

d) Removal of Network Non-compliance: In this last step, any misbehaving path is removed from usage in case non-compliant skews are detected between two nodes. To detect the misbehaving path, a node sends a packet that includes the estimated arrival time to the other node through the path. Given that the path is misbehaving, there is a node or attacker controlling a link that must be delaying the packets in an inconsistent manner. Therefore, as shown in our previous work [8], this non-compliant path must either eventually result in a time-out to maintain its slow skew or *acausally* delay packets to maintain its fast skew. In any case, a removal of the non-compliant path will be performed.

Lemma 4.1: Whenever an inconsistent wormhole attacker creates a non-compliant path, it can alter the skew of a path and remain undetected only for a finite time.

Proof: For a single-link non-compliant path, the proof can be found in [8]. The basic idea of the proof is that if the attacker-modified skew changes over time, then the inconsistent attacker can be detected. In order to construct a multi-hop algorithm able to detect an inconsistent wormhole attacker in finite time, we apply this basic idea.

Let us suppose there are two paths between two communicating nodes **A** and **B**, P_1 and P_2. P_1 has an aggregated skew approximately equal to the true skew; that is, $\hat{S}_{AB,P_1} \approx S_{AB}$ and $\hat{S}_{BA,P_1} \approx S_{BA}$. P_2 contains a malicious attacker, and its aggregated skew is different from the true skew; that is, $\hat{S}_{AB,P_2} \not\approx S_{AB}$ and $\hat{S}_{BA,P_2} \not\approx S_{BA}$.

To determine the non-compliant path, Node **A** sends packets to Node **B** alternately on the two paths exhibiting different measured skews, i.e., Node **A** sends packet n via Path P_1, then sends packet $n+1$ via P_2, and so on. Moreover, let us assume that the packets sent by Node **A** are periodic or equal duration apart; that is, the time duration between packet $n+1$ and n is equal to the duration between $n+2$ and $n+1$.

Given that $\hat{S}_{AB,P_1} \approx S_{AB} \not\approx \hat{S}_{AB,P_2}$, we examine the two cases where $\hat{S}_{AB,P_1} < \hat{S}_{AB,P_2}$ and $\hat{S}_{AB,P_1} > \hat{S}_{AB,P_2}$. For the first case, $\hat{S}_{AB,P_1} < \hat{S}_{AB,P_2}$, we expand both sides:

$$\frac{\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(n+1)}{\sigma_{AB,P_2}(n+3) - \sigma_{AB,P_2}(n+1)} > (1+\varepsilon)\frac{\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(n)}{\sigma_{AB,P_1}(n+2) - \sigma_{AB,P_2}(n)}.$$

Above σ_{AB} represents the transmitting timestamp, which is inserted by the sender, and the wormhole attacker cannot alter it without knowing the secret key used to authenticate the packet. Furthermore, given that the time durations between packets are equal, the denominators on both sides are equal. That is,

$$\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(n+1) > (1+\varepsilon)(\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(n)).$$

Solving the recursive functions we obtain,

$$\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(1) > (1+\varepsilon)^{(\frac{n}{2})}(\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(0)),$$

$$\rho_{AB,P_2}(n+3) - \sigma_{AB,P_2}(n+3) > (1+\varepsilon)^{(\frac{n}{2})}(\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(0)) + \rho_{AB,P_2}(1) - \sigma_{AB,P_2}(n+3).$$

Given that ρ_{AB,P_1} linearly increases with respect to n and $1+\varepsilon > 1$, there exists N such that for all $n > N$, for any defined time-out threshold T,

$$\rho_{AB,P_2}(n+3) - \sigma_{AB,P_2}(n+3) > T.$$

In other words, the path that contains an attacker would time-out and be removed in finite time .

For the second case, $\hat{S}_{AB,P_1} > \hat{S}_{AB,P_2}$, we expand both sides once more:

$$\frac{\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(n+1)}{\sigma_{AB,P_2}(n+3) - \sigma_{AB,P_2}(n+1)} < (1-\varepsilon)\frac{\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(n)}{\sigma_{AB,P_1}(n+2) - \sigma_{AB,P_2}(n)}.$$

Similarly,

$$\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(1) < (1-\varepsilon)^{(\frac{n}{2})}(\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(0)),$$

and there exists N such that for all $n > N$,

$$\rho_{AB,P_2}(n+3) - \rho_{AB,P_2}(1) < \hat{S}_{AB,P_1}\left(\rho_{AB,P_1}(n+2) - \rho_{AB,P_1}(0)\right).$$

In other words, the attacker would need to acausally forward packets and be removed in finite time ■

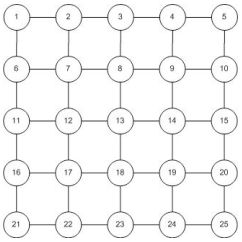
In case we only need an on-demand node-to-node synchronization, and not a network-wide synchronization, we eliminate the non-compliant links only along the paths connecting the nodes of interest.

V. Implementation Details

We have implemented our protocol on a testbed built using Crossbow Imote2 [11] sensors using the TinyOS 2.1 operating system [20] in order to verify its properties. Due to space limitations, we only highlight parts of the implementation and the main results of its evaluation. The implementation consists of the following major parts:

- Use of exponential smoothing to reduce quantization errors in timestamps and the drifts in clock skew over time.
- Generation of accurate timestamps, required to predict the remote node's clock time at which a packet will arrive.
- Use of a neighbor discovery protocol to permit the nodes to detect replayed packets.
- Exchange neighbor lists and reachable lists between nodes.

(a) Physical topology



(b) Link topology

Fig. 2. Experimental setup with 25 nodes

- Removal of inconsistencies discovered by the nodes.

Our implementation was tested on a 25 node network shown in Figure 2(a). In this figure there are 6 additional motes on the periphery that act as observers; and which log every packet they hear, for later analysis. Figure 2(b) shows the ID numbers of the nodes as well as their links.

Several experiments were performed, some of them with no attackers, and others with attackers. The latter resulted in inconsistent link behavior. We induced the nodes to think there is an attacker using two methods. We (i) made two nodes sharing a link exchange invalid skews between them, and (ii) made one of the neighbors of a given node A advertise an incorrect skew, resulting in non-compliances with respect to node A when other nodes in the network update their list of reachable nodes. In all the scenarios mentioned above and all the experiments performed, we obtained similar results:

- The product of the skews between nodes in the network stayed very close to 1, even over multi-hop links
- Whenever a compromised link is detected, nodes stop using it either because it is never advertised or because it is removed in the fourth step of our protocol, as mentioned in Section IV.

VI. Conclusion

In this paper, we have presented a secure network-wide clock synchronization protocol which at the same time allows for the secure discovery of the network topology. This network-wide protocol has been built on top of the scheme we proposed in a previous work [8] for a single-link scenario. The protocol is able to eliminate any inconsistent wormhole attackers, and even wormhole attackers that are consistent on a single-link basis, but not on a global basis, by performing a verification on every path of interest. Also, by disseminating the information of the links throughout the whole network, nodes are able to learn the topology of the network and remove any inconsistent links.

The protocol has been implemented on a Crossbow Imote2 sensor testbed, and we have verified the theoretical properties of the protocol in practice.

References

[1] D. L. Mills, "Internet time synchronization: The network time protocol," Network Working Group Request for Comments: 1129, pp. 1–29, Oct. 1989.

[2] K. Römer, "Time synchronization in ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing*. ACM Press, 2001, pp. 173–182. [Online]. Available: citeseer.ist.psu.edu/romer01time.html

[3] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[4] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems (SenSys-03)*. New York: ACM Press, Nov. 2003, pp. 138–149.

[5] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, Baltimore, MD, USA, Nov. 3-5, 2004*, J. A. Stankovic, A. Arora, and R. Govindan, Eds. ACM, 2004, pp. 39–49. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031501

[6] R. Solis, V. Borkar, and P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," *45th IEEE Conference on Decision and Control*, pp. 2734–2739, Dec. 2006.

[7] Y.-C. Hu, A. Perrig, and D. Johnson, "Wormhole attacks in wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 370–380, Feb. 2006.

[8] J. Chiang, J. Haas, Y.-C. Hu, P. Kumar, and J. Choi, "Fundamental limits on secure clock synchronization and detection of man-in-the-middle attacks," *Proceedings of the Twenty-Eight Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2009)*, April 2009.

[9] S. Graham and P. Kumar, "Time in general-purpose control systems: the control time protocol and an experimental evaluation," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, pp. 4004–4009 Vol.4, Dec. 2004.

[10] N. Freris and P. Kumar, "Fundamental limits on synchronization of affine clocks in networks," *Decision and Control, 2007 46th IEEE Conference on*, pp. 921–926, Dec. 2007.

[11] C. Technology, "Imote2 IPR2400 datasheet." [Online]. Available: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf

[12] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, 2005.

[13] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *Computers, IEEE Transactions on*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[14] 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: ACM, 2005, pp. 107–116.

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

[16] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*. New York, NY, USA: ACM, 2005, pp. 97–106.

[17] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 395–408, Feb. 2006.

[18] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 11-12, pp. 2314–2341, 2007.

[19] D. Bertsekas and R. Gallager, *Data networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[20] "TinyOS," http://www.tinyos.net/. [Online]. Available: http://www.tinyos.net/