Toward Clock Skew Based Wireless Sensor Node Services

Md. Borhan Uddin
Computer Science and Engineering
Polytechnic Institute of New York University
Brooklyn, NY 11201, USA
Email: borhan@cis.poly.edu

Claude Castelluccia
INRIA, Rhone-Alpes
655 avenue de l'Europe, Montbonnot
38334 Saint Ismier Cedex, France
Email: claude.castelluccia@inria.fr

***Abstract*—Clock skew is defined as the rate of deviation of a device clock from the true time. The frequency of a device's clock actually depends on its environment, such as the temperature and humidity, as well as the type of crystal.**

The main contributions of this paper are two-fold. First, we experimentally validate that MICAz and TelosB sensor motes have different and unique clock skews. Furthermore, the clock skew of a node can easily be monitored, even via a multi-hop Wireless Sensor Network (WSN). We argue that this feature can be used for identification of the nodes, detection of Wormhole and Sybil attacks.

Second, we show that the clock skew of a sensor node varies with the temperature. We explain how this property can be used to detect malicious and mal-functioning nodes and to geo-localize them.

***Index Terms*—Clock Skew, Covert Channels, Fingerprinting, Temperature, Malicious/Malfunctioning Mote Identification, Geolocation, Security, Wireless Sensor Network (WSN).**

I. Introduction

Clock skew has been used for remote fingerprinting of hosts on computer networks [22]. Variation of clock skew with respect to temperature has been used for revealing hidden services on Tor network [29]. So far so little has been known about clock skew on wireless sensor nodes and effect of temperature on the clock skew of sensor nodes.

In this paper, we tried to find out clock skew of different wireless sensor nodes with respect to a sink node and found that different motes have different clock skews. Clock skew varies a little from mote to mote and can be quantified with the unit 'parts per million' (ppm). This identity of motes based on different clock skew can be used for fingerprinting of nodes in WSN.

Clock skew of wireless sensor motes also varies with the variation of temperature [25]; this variation doesn't cause any problem of clock skew based mote fingerprinting, however. Variation of clock skew (a.k.a. clock drift) increases with the increase of temperature and decreases with the decrease of temperature. So, it seems that clock drift is a function of temperature. Thus estimating the variation of clock skew, we can estimate the current temperature of the end sensor nodes. Temperature data, sent from sensor motes having on board temperature sensor, can verified whether those motes are malfunctioning or lying based on the comparison of estimated temperature from variation of clock skew and received temperature data. Estimated temperature data, over the day and for long period, will also help us to estimate the geolocations of the nodes in WSN.

The end nodes of sensor network will send packets with timestamp to the sink node. Sink node will forward those packets to the measuring machine. Measuring machine will measure the clock skew of end nodes with respect to the clock of sink node. Sink node can also store the packets in its local storage and transmit later to the measuring machine. Measuring machine can be offline or thousands of miles away from the sink node. Sink node can be connected with measuring machine through the combination of GSM and Satellite network, Internet and other types of wired and/or wireless networks for monitoring clock skew from an office computer. Measuring machine only needs the packets from the sink node having timestamp of end nodes and sink node.

Our Contributions. We verified that clock skew based fingerprinting of sensor nodes can be successfully done on WSN. Clock skew based mote identification can be utilized for detecting Wormhole and Sybil attacks. Effect of temperature on the clock skew of sensor nodes can also be analyzed, which can be utilized for malicious or malfunctioning mote identification, estimation of temperature and geolocation of sensor nodes etc. Using 7 MICAz [6] and 2 TelosB [12] motes, we created single-hop and multiple-hop WSNs. We did experiment for both clock skew based fingerprinting of motes and analyzing effect of temperature on sensor motes by varying the temperature on wireless sensor motes. We implemented the synchronized timestamp sending mechanism as Zander et al. [35] on WSN and implemented the linear programming based skew detection mechanism as Moon et al. [28] using linear time algorithm of Dyer and Megiddo et al. [15], [26]. We showed that remote sensor mote fingerprinting can be done as remote physical device fingerprinting shown by Kohno et al.[22]. We experimented the variation of clock skew with the variation of temperature in wireless sensor motes as done by Murdoch et al. [29] on computer hosts for revealing hidden services in Tor network. We discussed the implication, pros and cons of clock skew based fingerprinting of sensor motes and analyzed the effect of temperature on the variation of clock skew on wireless sensor motes.

Organization. The rest of the paper is organized as follows. In section II we discuss about the clocks on almost all commercially available wireless sensor motes. In Section III we state the definition and formula of clock skew and different relevant terminologies as well as temperature variant terminologies of clock skew. In Section IV, we review the prior work related to clock skew, fingerprinting of sensor nodes and variation of clock skew with respect to temperature. In Section V, we describe our experimental setup for temperature invariant and variant experiments on clock skew. In Section VI, we present our experiments and results. In section VII, we discuss the applications of the results. In section VIII, we discuss the issues and limitations of the mechanisms. Finally, in Section IX, we discuss the conclusion and our future work.

II. Clocks On Wireless Sensor Motes

All of the commercially available wireless sensor motes have clocks constructed from hardware and software components. As hardware components, a microcontroller clock consists of an oscillator that ticks at a nominal frequency and a counter that counts the number of ticks. Microcontroller oscillators are of four different types-Crystal, Ceramic Resonator, RC(Resistor, Capacitor) and Silicon

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

Oscillators. Among these oscillators, crystal oscillators and ceramic resonators are more accurate and cheaper than other types. But they have the disadvantage of being sensitive to environmental conditions as electromagnetic interference (EMI), vibration, temperature, humidity etc. On the other hand, RC and Silicon oscillators are not accurate and have lower frequencies, although these are not sensitive to environmental conditions.

All of the commercially available wireless sensor motes are based on low power microcontrollers as- MICA and IRIS[5] family motes are based on Harvard architecture Atmel ATMega128 AVR microcontroller, Telos family motes are based on Texas Instruments MSP430 microcontroller and Intel Imote2[4] family motes are based on Intel PXA271microcontrollers [1]. From the datasheets of all the above mote families and their microcontrollers, it has been found that all of the microprocessors recommended external crystal oscillators for their timer circuit and all the motes use external crystal oscillators in their timer circuit.

So, the actual frequency of clocks of the motes depend on the environmental conditions (as electromagnetic interference-EMI, vibration, temperature, humidity etc.) and the other properties of the clock crystals.

Software components of the wireless sensor mote clocks are implemented in the operating systems installed on the motes. Almost all of the commercially available motes support open source TinyOS[13] operating system. TinyOS Extension Proposal (TEP) 102 [33] includes the implementation of different clocks and timer counter components of different sizes and frequencies for all the motes. In the implementation of these clock counters, it was essentially done-to divide or multiply the values of the hardware counters to get the desired frequency counters. Division or multiplications were done by the factors, which are equal to some power of 2, for simplicity and speed of computation. As a result the measured time by the clocks become coarse estimation of actual physical time. For example, clock crystals of TelosB motes have frequency 32768 Hz. For millisecond level timer, the frequency is divided by 32(=2^5); and thus 1024 milliseconds make one second by the TelosB motes clock. Similarly, MICAz motes have 28800 Hz clock; and thus 900 milliseconds make a second in the clock of MICAz motes.

For increasing the width of the software timer counter, rollovers of the smaller hardware counter were internally handled and updated in software counter. Among the implemented software timer counters in TEP102, width of the counters are 16-bit and 32-bit. Rollover time for the milli, micro and 32kHz timers for 16-bit and 32-bit counters as well as 48-bit and 64-bit counters (currently non existing in TinyOS implementation) are shown in Table I . As 32-bit counters were in rollover for Micro timer shortly and there was no implementation of 64-bit counter in TinyOS implementation, we implemented 64-bit counters from 32-bit counters for TMilli and T32kHz while handling the rollover internally. It would be better, if there exists a counter of 48-bit width; which would have rollover time about 3106 days for TMicro and longer rollover period for T32kHz and TMilli respectively as shown in Table I. It seems that 48-bit timestamp is preferable over 64-bit timestamp; because, timestamps will be transmitted in each packet; and thus low size of packet timestamp will decrease the packet size and increase the bandwidth in WSN.

For achieving longer battery life, most of the motes are designed to be in sleep mode when they are idle. In sleep mode or low power mode of all of the existing commercially available motes, timer counters are updated i.e., not stalled. Thus sleep mode doesn't cause any problem to get the timestamps from the motes for measuring clock skew.

III. Clock Skew and Temperature

A. Definitions

A clock is a piecewise continuous function that is twice differentiable. If $C(t)$ is the time reported by a clock at time t, there exists $C'(t) \equiv dC(t)/dt$ and $C''(t) \equiv d^2C(t)/dt^2$ for $t \geq 0$.

We used the following nomenclature from Moon et al.[28] to describe the clock characteristics. Let C_1 and C_2 be the two clocks:

offset: the difference between the time reported by two clocks. If time at two clocks C_1 and C_2 are $C_1(t)$ and $C_2(t)$ respectively, the offset of the clock C_1 relative to clock C_2 at time $t \geq 0$ is $C_1(t) - C_2(t)$.
frequency: the rate at which the clock progresses. The frequency at time t of C_1 is $C_1'(t)$.
clock ratio (α): the frequency ratio between two clocks. The ratio of C_1 relative to C_2 at time t is $\alpha = C_1'(t)/C_2'(t)$.
skew (δ): the difference in the frequencies of two clocks. The skew of C_1 relative to C_2 at time t is $\delta = C_1'(t) - C_2'(t) = \alpha \times C_2'(t) - C_2'(t) = (\alpha - 1) \times C_2'(t)$.
drift: The drift of clock C_1 relative to the clock C_2 at time t is $C_1''(t) - C_2''(t)$.

Clock skew is the difference between the clock frequencies of two clocks. When two clocks run at different frequencies, same physical time duration measured by two clocks will be different. When a delay measurement involves two clocks, the synchronization between those clocks has a tremendous impact on the accuracy of the measurement. For example, let us consider the case of measuring a packet delay between two nodes of WSN. The sending node adds a timestamp to a packet when it leaves the sending node, and the receiving node appends its receiving timestamp when it receives the packet. If the clocks of the two nodes are perfectly synchronized, the difference between the two timestamps is the end-to-end delay that the packet experienced in the WSN. If the clocks on the two nodes have a non-zero offset, but no skew, the difference between the two timestamps will be not only the end-to-end delay but also the offset. In one-way delay measurement, offset can't be distinguished from the measurement. If the clocks have a non-zero skew, not only is the end-to-end delay measurement off by an amount equal to the offset, but it also gradually increases or decreases over time depending on whether the sender clock runs slower or faster than the receiver clock.

B. Estimation of Clock Skew

Let us assume that measurer obtained a trace of N packets having the sending timestamps of end nodes and receiving timestamps of the sink node. Let us assume (as Figure 1) that-

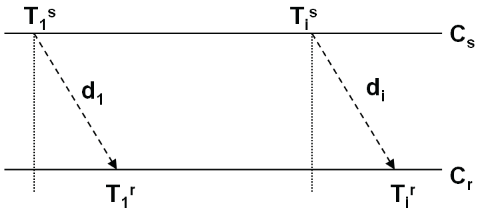

Fig. 1. Timing diagram showing equal delays (i.e., $d_1 = d_i$)

Sender(end node) clock C_s runs at frequency Hz_s
Receiver (sink node) clock C_r runs at frequency Hz_r
Sending and receiving timestamps of 1st packet are T_1^s and T_1^r
Sending and receiving timestamps of i-th packet are T_i^s and T_i^r
Propagation delays of 1st and i-th packets d_1 and d_i are equal (i.e.,

TimerCounter	16-bit	32-bit	48-bit	64-bit
TMilli	64 seconds	48.5 days	8716.3 years	571 million years
T32kHz	2 seconds	36.4 hours	272.4 years	17.8 million years
TMicro	64 milliseconds	1.14 hour	8.5 years	557844.6 years

TABLE I
ROLLOVER TIME FOR DIFFERENT WIDTHS (48-, 64-BIT: NON-EXISTING IN TINYOS) AND FREQUENCIES OF CLOCK COUNTERS

$d_1 = d_i$)
Elapsed sending time of i-th packet measured by C_s is $t_i^s = (T_i^s - T_1^s)/Hz_s$
Elapsed receiving time of i-th packet measured by C_r is $t_i^r = (T_i^r - T_1^r)/Hz_r$
Offset of i-th packet $\tilde{o}_i = t_i^s - t_i^r$
Now, we have N pairs of $(t_i^r$,$\tilde{o}_i)$ for $i \in \{1, \dots, N\}$.

As Kohno *et al.* [22], for estimating the skew, we draw a line above all offset measurements and while minimizing the sum of distance between the line and the offset points. By applying the linear programming algorithm of Moon *et al.* [28] the above line is derived. This finds an estimate of the upper bounds of linear offset components- $\hat{o}(t) = \hat{\delta} \times t + \gamma$ while minimizing the following expression:

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{o}(t_i^r) - \tilde{o}_i)$$

where $\hat{\delta}$ is the estimate of the skew.

We used linear time algorithm as [15], [26] to estimate the line using the linear programming approach.

C. Analysis of Effect of Temperature

In the previous section, we discussed the techniques to estimate the clock skew i.e., we didn't yet discuss about the measurement of temperature variant part of clock skew. Murdoch [29] first time did the experiment for correlating temperature with change of clock skew. As Murdoch, we removed constant skews from delay offsets, resulting the variable offsets. We estimated the $Denoised\ (t_i^r, \tilde{o}_i - \hat{o}(t_i^r))$ for $i \in \{1, \dots, N\}$ based on a sliding window. In figure 8 upper blue curve is the denoised form of the green dots (i.e.,$(t_i^r, \tilde{o}_i - \hat{o}(t_i^r))$). Denoising is done based on a sliding window and multiple passes of the sliding window are used from different starting positions to make the denoised curve smooth. Final denoised curve is constructed taking the average of all generated denoised curves generated in different passes. $Denoised\ (t_i^r, \tilde{o}_i - \hat{o}(t_i^r))$ is differentiated based on a differential sliding window to find out the drift for $i \in \{1, \dots, N\}$. Drift is plotted against the plotting of temperature. Figure 9 shows the plotting of drift against the plotting of temperature. Denoising is done as first derivative of variation of clock offset and drift is derived from differentiation of denoised offset; thus 'drift'- second derivative of clock offsets, matches with the nomenclature of Moon *et al.* [28].

IV. BACKGROUND AND RELATED WORK

There exists a significant amount of prior work on clock skew based host identification in computer networks and usage of effect of temperature on clock skew of hosts in computer networks. But it seems that our one is the first in clock skew based mote identification and finding effect of temperature on mote clock skew in WSN.

It has long been known that seemingly identical computers have different clock skews and clock skew varies with temperature. The NTP [27] specification states the method of reducing clock skew by short periodic synchronization with time servers, although synchronized machines have clock skew between the intervals of synchronizations.

Paxson [31] initiated a line of research geared toward eliminating clock skew from network measurements, and the linear programming algorithm for clock skew measurement we use is based on a descendent of the Paxson paper by Moon et al. [28]. Linear time algorithm of the above linear programming approach is solved by the linear time linear programming algorithms of Dyer [15] and Megiddo [26].

Kohno et al. [22] used timing information from remote computers to fingerprint their physical identity. By examining timestamps from different machines they estimated the clock skew of the machines, showed that different machines have different clock skew and it might be a viable means of identifying physical machines based on clock skew.

Murdoch [29] first showed clock skew varies with variation of temperature on network hosts. He used this property of clock skew to identify hidden services in Tor network. He showed that three collaborating servers providing a hidden service on Tor network can be easily identified by receiving timestamps from candidate servers in Tor networks. As three servers in Tor networks sharing same service will have similar load which will lead to similar type of temperature change, thus similar change in clock skew as well. Therefore three servers serving on same hidden service in Tor network can be identified and thus anonymity of Tor network can be compromised.

Zander et al. [35] devised an improved clock skew measurement technique based on synchronized sampling of clocks. He shows that synchronized sampling of clocks reduces the quantization error and makes the estimated clock skew more accurate. Synchronized sampling also helps in measuring clock skew for low resolution timestamps (e.g., HTTP servers having Hz level clocks thus timestamps in second level resolution).

Recently, Huang et al. [21] proposed clock skew based node identification in WSN using Flooding Time Synchronization Protocol (FTSP). In their work, they didn't implement the measuring channel as a covert channel, however. They proposed to measure and retain the clock skew of the node on the node itself and later transmit the calculated skew to the sink. Apparently, this method of clock skew estimation using FTSP is coarse estimation of skew based on current offset and previous skew, which rules out the suitability of the approach for identification of large number of sensor motes in WSN. Clock skew computation on mote isn't feasible because, sensor motes don't have enough on board storage for holding large number of time offsets and enough computational capabilities to do the costly linear programming based skew estimation. Moreover, as their clock skew estimation channel isn't covert, their approach might have security vulnerability issues for detecting Sybil attack [30].

Wireless device fingerprinting based on transmission medium of the devices have been performed on a number of devices with different transmission media as Bluetooth-enabled mobile phones [20], 802.11 wireless cards [18] and Chipcon 1000, 433MHz radios [32].

V. EXPERIMENTAL SETUP

We had two different experimental setups- first one to test whether different motes have different clock skew and second one to test whether clock skew varies with variation of temperature. In first setup, if different motes have different clock skew with respect to the sink node, we will be able to identify the motes based on their clock skew; i.e., clock skew with respect to the sink node of a mote will

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

be the physical identity of the mote. In the second setup, if clock skew varies with variation of temperature, we will be able to find out the relation between variation of clock skew and temperature change.

In both the setups, we did our experiments for single and multiple hops of wireless sensor network with two different types of commercially available and most common wireless sensor motes- MICAz [6] and TelosB [12] motes. MICAz is based on Atmel AVR Atmega128 8-bit microcontroller and TelosB is based on Texas Instruments MSP430 16-bit microcontroller. MICAz has 51-pin expansion slot to connect with computer through MIB [14] serial interface and TelosB has USB interface to connect directly with computer. Both the motes run on open source TinyOS [13] operating system and operate over 2.4 GHz IEEE 802.15.4 protocol [3] compliant to Direct Sequence Spread Spectrum (DSSS) radio. We used a total of 9 motes (7 MICAz motes and 2 TelosB motes) to run our experiments.

Sink node was connected through MIB510 [14] serial interface with the RS232 serial port of a desktop computer running on open source OS Ubuntu and TinyOS 2.1.0 package was installed on that machine. Serial packet sniffer TinyOS 2.1.0 *MsgReader* is used to capture the transmitted packets from the sink node over the serial interface.

A. Single-hop WSN Setup

In single hop WSN setup, one sink node was connected with desktop computer and rest of the nodes acted as end nodes. End nodes only transmitted packets with transmitting timestamp to sink node over the wireless channel after certain intervals. Sink node received packet over the wireless channel, appended its receiving timestamp and transmitted it instantaneously to the desktop computer over the serial interface. Figure 2 shows our setup for single hop wireless sensor network.

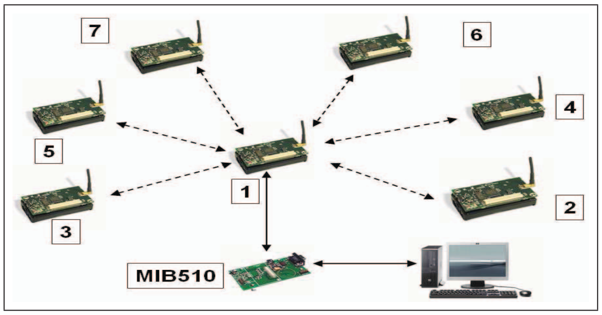

Fig. 2. Single-hop WSN Setup with 7 MICAz motes. Node 1 is the sink node connected with computer through MIB510 serial interface. Nodes 2-7 are acting as end nodes and sending packets to sink over the wireless channel.

B. Multiple-hop WSN Setup

In multiple-hop WSN setup, there were 3 types of nodes - sink, forwarder and end nodes. End nodes only transmitted the packets with sending timestamps over the wireless channel after certain intervals. Forwarding node captured them and forwarded the packets to sink or another forwarder towards the sink node over the wireless channel without modifying the content of the packets. Sink node captured all the packets over the wireless channel transmitted to it and appended its receiving timestamps on each packet and forwarded the packets to desktop computer over the serial MIB510 interface. Timestamps were only sent from end nodes to sink. Sink didn't send any timestamps to end nodes. Sink to end nodes paths are only used for sending request from measuring machine through sink node to start and stop of transmission of packets and transmission parameters- as packet transmission rate etc.

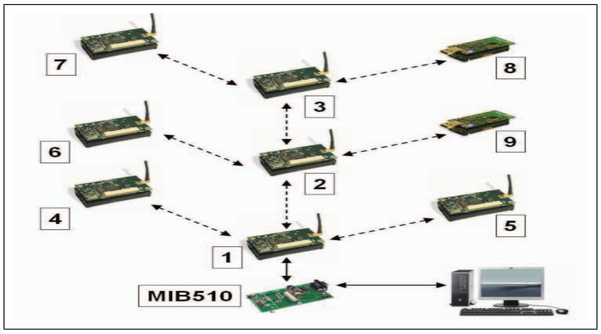

Fig. 3. Multiple-hop WSN Setup with 7 MICAz (nodes 1-7) and 2 TelosB motes (nodes 8-9). Node 1 is the sink node. Nodes 2 and 3 are forwarding nodes on WSN and remaining nodes are end nodes.

Figure 3 depicts our setup for multiple-hop WSN with fish-bone topology. Nodes having node id 1-7 are 7 MICAz motes and nodes having node id 8-9 are 2 TelosB motes. Node id 1 is the sink node, attached with computer through MIB510 Serial interface. Nodes 2 and 3 are forwarding nodes and nodes 4-9 are six end nodes. So, we have 4 MICAz (nodes 4-7) and 2 TelosB (nodes 8-9) motes as end nodes. Multiple hops of sensor network were implemented in an office building placing different-hop motes in different rooms of the building. Forwarders were placed on corridor of the building.

C. Initialization of motes with nesC TinyOS Applications

Transmitter, forwarder and sink node functionalities were implemented in TinyOS[23] environment using nesC[16] language. Programs for different functionality were installed on corresponding motes. TinyOS packets don't have any built-in timestamps as computer networks packet timestamping feature (e.g., TCP, IP and HTTP packet timestamps). We had to implement the tinyOS packet timestamping feature on applications to add sending and receiving timestamps on packets. Along with timestamps, sending node id, sending packet sequence number, current temperature etc. were sent from the end nodes. Sink node appended its node id, receiving timestamp and packet count number, temperature and forwarded the packets to the desktop computer over the serial interface.

D. Setup for Clock Skew Measurement in Constant Temperature

In this setup, temperature of the sensor mote environment was not changed. Packet transmission and capturing were done in normal room temperature. Both single- and multiple-hop WSN configurations were used to capture the data for fingerprinting of wireless sensor motes. Micro and 32kHz level timestamps were used in single-hop WSN (using only MICAz) and only 32kHz level timestamps were used in multiple-hop (using both MICAz and TelosB) WSN. Both the types of timestamps were used to test the effect of precision (frequency) of timestamps on skew calculation; and both types of motes were used to test the skew calculation on sensor network with heterogeneous motes.

E. Setup for Clock Skew Measurement Varying Temperature

For testing the effects of temperature on mote clock skew, temperature of end nodes were varied. Firstly, the motes were set to room temperature for certain period to get the constant skew. After that temperature of motes were changed by heating up and cooling down the mote environment. We used air heater to heat up and cooler to cool down the motes. Maximum heating temperature was about 55-60 degree celsius and minimum cooling down temperature was 4-5 degree celsius. Highest temperature of the experiment is approximately near to the ever reported highest temperature (136 degree Fahrenheit i.e., about 58 degree centigrade) on the Earth [8].

Both MICAz and TelosB motes were used to run this experiment so that effect of temperature on both of them can be compared. Both the single-hop and multiple-hop WSNs were used to test the effect of temperature on mote clock skew. Same thermal conditions were applied at a time to the motes on the same hop count i.e., all the motes within same hop are heated up or cooled down simultaneously.

For collecting the temperature values from the motes, we used interfaces of motes for collecting temperature. MICAz motes don't have built-in on board temperature sensor but TelsoB motes have built-in onboard temperature sensor. For collecting temperature values from MICAz motes we attached MTS300 [7] sensorboards on MICAz motes. Four MTS300 sensorboards were attached with four end nodes MICAz motes (motes 4-7 of figure 3) on their on board 51-pin expansion connectors. MTS300 sensorboard has light sensor, temperature sensor, microphone and buzzer on it. In our installed application on MICAz motes, we implemented the functionality of reading value of temperature sensor on MTS300 sensorboard and sending the temperature values on the transmitted packets. The temperature sensor on the MTS300 board is a thermistor (i.e., changes resistance with change of temperature) . Thermistor readings (extracted from the received packets) are converted to celsius values in the desktop computer using the following formula for MTS300 available at [7]:

$$\frac{1}{T(K)} = a + b \times ln(R_{thr}) + c \times [ln(R_{thr})]^3$$

and $T(C) = T(K) - 273.15$; where, $R_{thr} = \frac{R1 \times (ADC\_FS - ADC)}{ADC}$, $a = 0.00130705$, $b = 0.000214381$, $c = 0.000000093$, $R1 = 10k\Omega$ $ADC\_FS = 1023$; $T(K)$,$T(C)$= Temperature in Kelvin,Celsius respectively; ADC = output value from Mote's ADC thermistor measurement.

Two TelosB motes have built-in on-board temperature, humidity and light sensors. Temperature sensor on TelosB is Sensirion SHT11 [2], [10] which is also a thermistor. In the application installed on the two TelosB motes, thermistor value is read and wrote to each transmitted packet. Thermistor values (extracted from received packets) of Sensirion SHT11 are converted to celsius temperature in the measuring desktop computer using the following formula available at [10] and also in TinyOS 2.1.0 *SensirionSht11C* component documentation:

$$T(C) = -38.4 + 0.0098 \times data;$$

where, $T(C)$= Value of temperature in Celsius; $data$= output value from Mote's ADC thermistor measurement.

VI. Experiments and Results

The experiments were conducted with two main goals- firstly, to find out whether the motes can be identified with their clock skew and secondly, finding out the relationship between temperature and change of clock skew on sensor nodes.

A. Estimation of Clock skew without varying Temperature

In single hop WSN setup, six MICAz motes were used as end nodes and one MICAz mote was used as sink node. End nodes sent packets to sink node after 4096 milliseconds synchronized intervals. Both Micro and 32kHz timestamps were sent from the end nodes in single-hop WSN setup but only 32kHz timestamps for multiple-hop setup. The test was run for about 10K seconds. From the sniffed packets offset have been calculated and plotted in graph. Linear time linear programming based skew estimation algorithm has been used to calculate the skew as Moon et al.[28]. A line has been drawn above all the points of clock offset as the estimation of skew. The distance between the line and offsets have been kept minimum as well to satisfy the linear programming approach of Moon et al. Linear time algorithms of Dyer [15] and Megiddo [26] have been used for the linear programming approach.

For both Microsecond and 32kHz level timestamps we did the skew estimation. 32khz timestamps are later converted to millisecond level timestamps dividing by 32 for convenience of comparison between micro and millisecond level timestamps. Table II shows the calculated skew for the end nodes using 32kHz and microsecond level timestamps. From the table, it has been found that calculated clock skews using both the timestamps are almost same for all the nodes.

Mote	Clock Skew (T32kHz)	Clock Skew (TMicro)
Node 2	-16.7375 ppm	-16.7359 ppm
Node 3	3.8091 ppm	3.8054 ppm
Node 4	-10.6432 ppm	-10.6398 ppm
Node 5	-21.1137 ppm	-21.1148 ppm
Node 6	3.0436 ppm	3.0517 ppm
Node 7	7.8206 ppm	7.8194 ppm

TABLE II
Estimated Clock Skew using 32kHz and Microsecond level timestamps for single-hop WSN using MICAz motes as of setup Figure 2

It has been found that using both micro and millisecond level timestamps, skew can be estimated.

From Table II, it is clear that for single hop WSN it is possible to identify the sensors based on clock skew. So, sensor mote fingerprinting seems feasible for single hop WSN and both Micro and 32kHz level timestamps can be used to identify the clock skew.

Similar type of experiment as above has been done for multi-hop WSN setup as figure 3. Packets from end nodes 4-9 have been sent to desktop computer by sink node for about 7K seconds and skew is calculated as single-hop setup using linear time linear programming approach. Skew of the four MICAz motes with respect to a sink MICAz are plotted in figure 4. It has been found that nearby nodes (few hops away from sink node) have lower variation in delay offset than the nodes far from the sink node. i.e., nodes 4 and 5,which are single hop away, (red and green delay offsets) have lower delay offsets than nodes 6(two-hop away) and 7 (three-hop away). But skew is yet clearly identifiable and different motes have different skew on multiple-hop WSN. So, from the above result it is clear that in multiple-hop wireless sensor network of similar type of motes, clock skew based sensor identification is feasible.

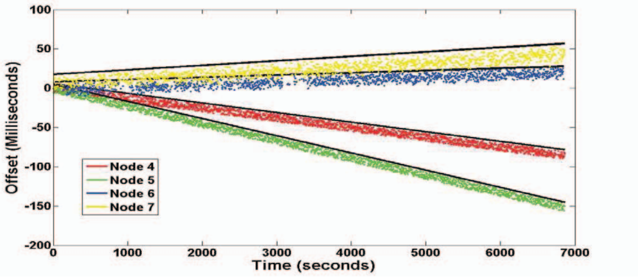

Fig. 4. Clock skew of MICAz Motes in Multi-hop WSN using 32kHz level TimeStamps

TelosB and MICAz motes have different measurements of same physical time. In TinyOS implementation of TelosB mote Local Time count, 1 second is taken as 1024 milliseconds. But in tinyOS implementation of MICAz, 1 second is taken as 900 milliseconds.

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

As a result TelosB end nodes have very large skew with respect to MICAz sink node. For plotting them on same graph offset of TelosB motes are scaled down with the factor $900/1024$. Figure 5 shows the clock skew of 4 MICAz motes (nodes 4-7) and two TelosB motes (nodes 8-9). It is found that motes with smaller hop count have smaller delay offset than motes with larger hop count. TelosB motes seem to have less delay because of scaling down the offset value with the factor $900/1024$. Although with the increase of hop count delay offset increases, it seems obvious that different motes (whether same or different types) have different skews in multiple-hop WSN.

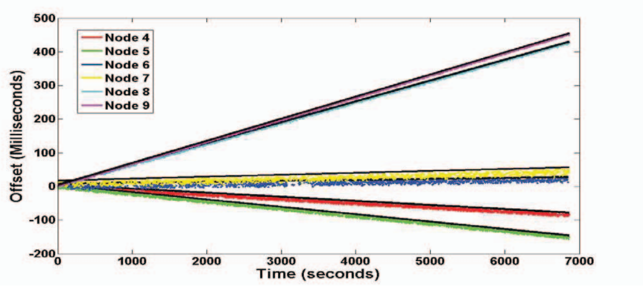

Fig. 5. Estimated clock skew of MICAz and TelosB Motes in multiple-hop WSN using 32kHz level TimeStamps. Nodes 4-7 are MICAz motes and nodes 8-9 are TelosB motes.

Table III shows that different motes (similar or different types) have different clock skews.

Mote	Clock Skew (T32kHz)
Node 4 (MICAz)	-11.3785 ppm
Node 5 (MICAz)	-26.1601 ppm
Node 6 (MICAz)	7.6290 ppm
Node 7 (MICAz)	11.9188 ppm
Node 8 (TelosB)	126.7999 ppm
Node 9 (TelosB)	118.9366 ppm

TABLE III
ESTIMATED CLOCK SKEWS (USING 32KHZ LEVEL TIMESTAMPS) OF MULTIPLE HOPS MICAZ AND TELOSB MOTES AS OF SETUP FIGURE 3

From the results, it seems clear that different motes (similar or different types) have different clock skews and it can be measured both in single and multiple-hop WSNs with micro, 32kHz or millisecond level timestamps.

B. Effect of Temperature on Clock Skew

For experimenting the effect of temperature on mote clock skew, packets were captured from multiple-hop WSN as setup of Figure 3 and varying the temperature on end nodes. End nodes of same hops are kept in same thermal condition while running the experiment. Temperature on end nodes were varied by heating up and cooling down the surrounding air of the nodes using air heater and cooler. Traces for about ten thousand seconds are taken to analyze the thermal effect on clock skew. Figure 6 shows the estimation of clock skew in multiple-hop WSN while varying the temperature from 4 to 60 degree celsius. It seems that clock skew varies a little with variation of temperature and yet different motes have different clock skews. So, variation of temperature doesn't affect clock skew based fingerprinting.

Effect of temperature on clock skew was analyzed by doing the following steps:

1) Step 1: Estimation and Removal of Constant Clock Skew: From sending and receiving timestamps of all of the packets, constant clock skew of each of the motes were determined using linear programming skew estimation algorithm as [28]. The upper black straight line was drawn above all the green offset points while minimizing the distance between the points and the straight line using linear time [15], [26] linear programming algorithm [28]. It has been found that after varying temperature yet constant skew remains dominant over variable skew. Figure 6 shows the clock skew of four MICAz motes (nodes 4-7) while varying the temperature from 4 to 60 degree centigrade.

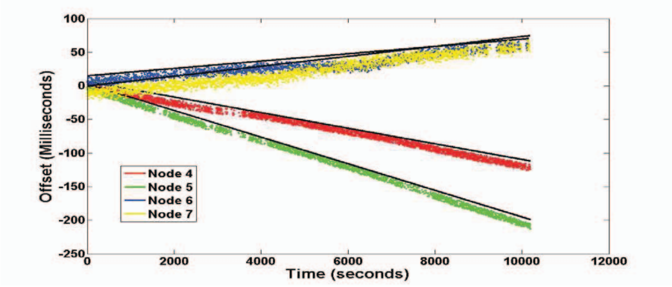

Fig. 6. Estimated clock skew of 4 MICAz motes in multiple-hop WSN while varying the temperature (from 4-60 degree centigrade). Colored points show the offset values of corresponding motes and black straight lines above the offset values show the estimated clock skews

Figure 7 shows the estimated constant clock skew (upper black straight line) component calculated from delay offset. Corresponding temperature data received from attached or onboard temperature sensors are plotted as red curve. Constant skew components were removed from the actual delay offsets by subtracting the estimated constant skew offset (i.e., values on skew straight line) from the actual offset values (corresponding green dots). Resultant variable skew components are plotted in figure 8 (green dots) against the corresponding temperature (red curve).

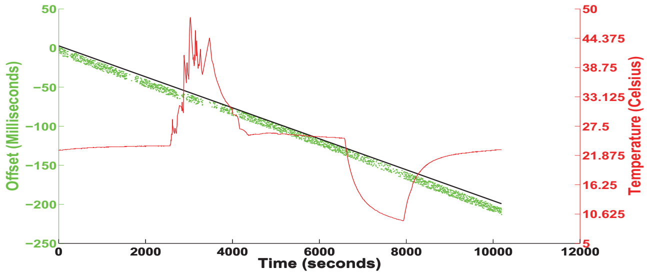

Fig. 7. Estimation of Constant Clock Skew from the offset values. Green dots indicate offset values and Upper black straight line shows the estimated clock skew

2) Step 2: Denoising the Variable Skew: Variable skew components were denoised using multiple-pass sliding window based linear time [15], [26] linear programming algorithm as Moon et al.[28]. The final denoised line(upper blue line of Figure 8) is the average of denoised lines for all of the passes. Window size is an important factor for denoising; and possibly it is a function of total data points and expected variation of temperature. Several passes with several different initial sliding window positions also make the denoising more accurate. We used a total 8 windows and 3 passes of the windows over all the data points for denoising the variable skew in Figure 8. Final denoised blue line is the average of the 3 denoised lines produced in the 3 different passes.

3) Step 3: Finding out the Drift from De-noised Variable Skew: Drift is calculated by taking first derivative of denoised variable skew curve (i.e. upper blue curve of Figure 8). Drift is plotted on Figure 9 as dashed pink line against the corresponding temperature values(red curve). The first order derivative on each data point of denoised variable skew curve is calculated based on forward, backward and central differentiation methods with all neighboring

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

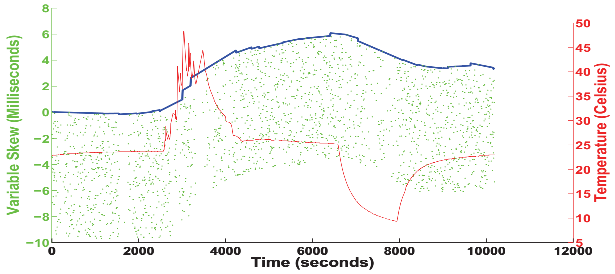

Fig. 8. Denoising of Variable Skew using a total of 8 windows and 3 passes

points with respect to a maximal differential window size. The maximal differential window size is also an important factor to get the actual drift. Size of the maximal differential window is possibly a function of total data points and expected variation of temperature. For finding the drift of denoised curve of Figure 8, we used a total of 12 maximal differential windows.

Scaling and shifting of the drift curve is done to find out matching of it with temperature curve. Scaling and shifting coefficients are adjusted on trial-and-error basis. These scaling coefficients seem to be constant over time for a particular end node and sink node pair.

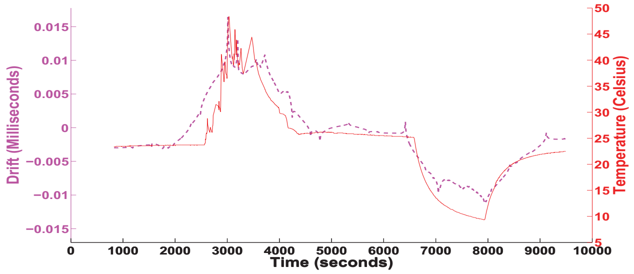

Fig. 9. Estimated Drift (pink dashed line) and Measured Temperature (red line)

From the plotted drift graphs against temperature as Figure 9 for all end node MICAz motes, it has been found that drift varies with variation of temperature. With increase of temperature drift increases and with decrease of temperature it decreases; thus, drift seems to be a function of temperature.

Above steps 1-3 are done for all end nodes (i.e., nodes 4-9 of Figure 3). It has been found that total number of hops between the end node and sink doesn't affect much on the matching between drift and temperature for all of the motes.

TelosB motes differ with MICAz motes in matching of drift with temperature. Drift of MICAz mote increases with increase of temperature and decreases with decrease of temperature. For increase of temperature over the room temperature drift of TelosB mote matches with temperature, but for decrease of temperature from room temperature its drift has the reverse effect- it increases instead of decrease. Figure 10 depicts the result. Upper graph is for a MICAz and lower graph is for a TelosB mote. It seems that upper graph for MICAz well matches with temperature while lower graph for TelosB matches at the increased and normal room temperature. But for decrease of temperature below room temperature it doesn't match. It has the compensation for lower temperature than room temperature, although the compensation is too over. Possibly, it is difficult to compensate accurately for temperature change and TelosB motes missed to add any compensation for temperature above the room temperature.

VII. Applications

Based on the results obtained from the above experiments, we discuss the following facts and findings.

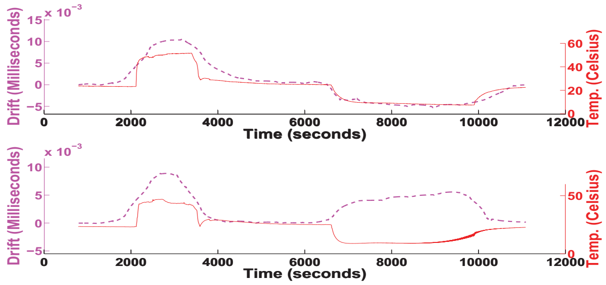

Fig. 10. Estimated Drift (pink dashed line) of a MICAz mote (upper graph) matches with measured temperature (red line) and Drift of a TelosB (lower graph) mote matches with temperature for higher than room temperature but for lower temperature it is overcompensated.

A. Fingerprinting (Identification) of Wireless Sensor Motes

From the Tables II and III and Figure 5, it has been found that different motes have different clock skew. Clock skew can be measured both in single hop and multiple-hop WSNs with both milli and micro level timestamps. Clock skew varies from mote to mote with parts per million (ppm) level and remains constant over time for a particular mote. From the table III, it has been found that TelosB motes have significantly higher clock skew than MICAz motes. So, clock skew can be used to identify the type of motes. From the Table II, it has been found that each of the six MICAz motes has distinct ppm level skew than other MICAz motes. From the table III, it has been found that each of 4 MICAz motes have distinct ppm level clock skew and two TelosB motes also have distinct clock skews. So, clock skew can be used to identity the motes of similar types and models as well as different types of motes.

Currently most wireless devices, including WSN motes, use physical MAC address as their ID. But it has been found that spoofing of the wireless MAC address is easy [34]. So, clock skew can be used along with wireless MAC address for bootstrapping physical identification of wireless sensor motes.

B. Detection of Virtual Sensor Nodes, Wormholes and Sybil Attack in WSN

As the end nodes only send the timestamps to the sink node and sink node doesn't send any timestamps to the end nodes, it is difficult for end nodes to measure and alter the clock skew. Clock skew will be measured using a covert channel in WSN. End nodes won't be aware of the clock skew measurement by the measuring machine or by the sink node. End nodes won't be able to distinguish between the packets which are used for clock skew measurement and which are used for other time critical applications in WSN.

End nodes will be functioning with many other time sensitive applications and their Quality of Service (QoS) will also depend on timestamps. Packets will be dropped for too old or future timestamps in intermediate forwarding nodes and QoS in WSN will also depend on the timestamps. So it is hard for motes to lie about their current time on their clocks. To change clock skew, an attacker needs to get timestamps from both the sink node and end nodes. Packets with both the timestamps are available on the sink to measuring machine link (serial connection with MIB510 on experimental setup- but it might be replaced with variety of network connections as GSM, Satellite and computer networks for remote monitoring of WSN from an office computer). If the sink to measuring machine link is secure (through encryption or any other means of physical layer security), it will be very difficult for an attacker to know about the timestamps and thus alter the clock skew. So, it seems that clock skew of motes is hard to spoof in WSN. Gehani et al. [17] proposed authenticated exchange

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

of sensor node timestamps, which can be used for tamper resistance of the transmitted timestamps in WSN.

Clock skew depends upon the physical clock of the mote. If a malicious mote broadcasts packets with multiple fake wireless MAC address, i.e., posing as many virtual motes, it will be easy for the measuring machine to detect the virtual motes. All of the virtual motes will have same clock skew as they have same physical mote clock. End nodes of wormhole links [24] can be identified using clock skew based mote identification; as end nodes of a wormhole link also relay timestamp of same physical clock. Similarly, Sybil attack [30] can be detected by physical clock skew based identification of wireless sensor motes.

C. Malicious/Malfunctioning Node Identification

Clock drift seems to be a function of temperature, as it has been shown in Figure 9. Clock drift increases with increase of temperature and it decreases with decrease of temperature. If a mote doesn't have on board temperature sensor, yet we can estimate the temperature of that node based on its clock drift.

If the mote has an on-board temperature sensor and it sends temperature data to the sink, we can validate the credibility of reported temperature by plotting the temperature data with the clock drift. In Figure 9 and 10, we did some scaling and shifting of the drift curve (dashed pink curve) to match it with temperature curve. These scaling coefficients seem to be constant over time for a particular sensor node. Thus if we know these coefficients of the mote, we can verify the temperature data obtained from the mote.

These scaling and shifting coefficients must be known by the end node to adjust the timestamps for lying about its temperature and remain undetected. These coefficients also might be related with sink's timestamps. If sink's timestamp becomes un-available to end nodes, it might be difficult to decide about these coefficients. Clock drift is calculated as the first derivative of the variable clock skew. As clock skew seems hard to alter for end nodes, first derivative of clock skew i.e., clock drift will also be hard to alter for end nodes.

If the mote reports more or less drift than its temperature (might be obtained from some other reliable sources of temperature), we might identify that the mote is malfunctioning or lying about its timestamps. If the mote has onboard temperature sensor, it can add compensation for the temperature change to adjust its clock skew. For example, in lower graph of Figure 10 TelosB mote added skew compensation for temperature below the room temperature; although it has been overcompensated. If the mote doesn't have onboard temperature sensor, it isn't possible for mote to add thermal compensation for clock skew or lie about its clock skew and clock drift. Thus, clock skew based malfunctioning mote identification becomes feasible.

D. Geolocation

Gupchup et al. [19] proposed 'Sundial'- an on board opto-electric sensor based geolocation estimation in WSN by measuring day light. Temperature varies with the variation of day light. So, similar geolocation estimation technique can be employed by replacing day light measurement using opto-sensor with estimated temperature from clock drift.

Clock drift changes with change of temperature. Clock drift seems to be a function of temperature. Temperature can be estimated by monitoring the clock drift. This property can be used for estimation of geolocation of sensor nodes. Longitude of a place can be estimated by measuring daily peak temperatures to establish local time. Latitude can be estimated by measuring change of day lengths over a reasonably long period of time. This process of determining geolocation requires the sink node to be kept in conditioned (fixed) temperature, which seems feasible to achieve for a single sink node in WSN.

VIII. Issues and Limitations

A. Remote Monitoring of Clock Skew and Effect of Temperature

Both clock skew and effect of temperature on clock skew can be remotely monitored i.e., sink node doesn't need to be directly connected with a computer. For example, in Figure 2 and 3, sink node (node 1) can be connected with computer through combination of GSM, satellite and computer networks instead of direct connection with computer through MIB510 serial interface. Thus, both the clock skew and drift of remote sensor nodes can be monitored from an office computer, which might be thousands of miles away from the WSN. Packets containing timestamps of end nodes and sink node are sufficient to monitor the clock skew of end sensor nodes and effect of temperature on their clock skew. Distance between the sink node and the measuring computer doesn't have any effect on clock skew measurement; because, only the timestamps of end nodes and sink nodes are used to measure the clock skew. Timestamps of intermediate forwarding nodes and measuring computer aren't used to measure the clock skew of the wireless sensor nodes. For correct measurement of effect of temperature on clock skew, it must be ensured that sink node is in conditioned temperature.Its temperature mustn't change with the change of temperature of outside environment; which is unlike the end nodes as end nodes won't be in conditioned temperature. As Murdoch [29], we took temperature as dominant factor; i.e., we didn't test the clock skews and drifts by varying other factors as humidity, vibrations and electro-magnetic interferences (EMI) etc., however.

B. Thermal Compensation on Clock Crystals

Thermal compensation can be added to clock crystals of wireless sensor motes. However, adding thermal compensation isn't easy to clock crystals. Either it becomes over or under-compensation. Moreover thermal compensation requires an onboard built-in temperature sensor on sensor mote. For example, in Figure 10 we have found that thermal compensation has been added to TelosB motes for temperature below the normal room temperature but it has been overcompensated.

Currently Temperature Compensated Crystal Oscillators (TCXOs) [11] and Oven-Controlled Crystal Oscillators (OCXOs) [9] are commercially available. But these crystal oscillators are used for mission critical purposes and are too costly to afford on low cost wireless sensor motes.

Usage of these crystal oscillators will have effect on the analysis of effect of temperature on clock skew but won't have any effect on clock skew based fingerprinting of wireless sensor motes. Because, thermal compensation doesn't alter the distinctness property (i.e., different skew) of clock crystals.

C. Selection of Parameter Values for Analysis of Effect of Temperature

In analysis of the effect of temperature on clock skew (in steps 2 and 3 of subsection VI-B), it seems that selection of total number of Windows or Window sizes and selection of number of passes seem open problems. In step 2 (Figure 8), we have to select total number of denoising windows (or window size- which can be calculated from the total number of windows and vice versa) and number of passes of the windows over the data points. If window size is too small (i.e. total number of windows are large), the estimated denoised curve becomes too zigzag and actual denoising isn't achieved. On the

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503

other hand, if denoising window size is too large, it does the coarse estimation of actual data points which also doesn't reflect the actual scenarios of temperature changes. Similarly, differential window size also has the same effect on step 3 (Figure 8). Total number of passes of the windows over the data points have effect on smoothing the drift curve and reflecting the scenarios of temperature changes more accurately. Large number of passes make the estimation more accurate but execution time increases proportionately. So, success of this approach depends on the finding out optimal window sizes (or number of windows) and total number of passes. Incorrect choice of window size and number of passes for steps 2 and 3 will make the correlations difficult to observe. In step 2 and 3 of subsection VI-B, we selected the number (or size) of windows and number of passes in trial-and-error basis.

IX. Conclusion and Future Work

In this paper, we explored the clock details of different commercially available wireless sensor motes. We showed that different motes have different clock skew and motes can be identified by their ppm level clock skew. This property can be used for clock skew based fingerprinting of wireless sensor motes, identification of virtual motes on virtual honeynets, wormhole and sybil attack detection in WSNs. We also showed that clock skew varies with the variation of temperature on wireless sensor motes, which can be used for malicious and malfunctioning mote identification, estimation of temperature and geolocation of sensor motes in WSN. We discussed different issues, pros and cons of the two methods. In our future work, we will try to explore more on feasibility and mechanism of clock skew based geolocation estimation in WSN. We will also try to explore feasibility and security aspects of clock skew based wireless sensor mote identification instead of currently used MAC address based mote identification- which is easy to spoof [34] and find out feasible mechanism for malicious (lying on its temperature/location) mote identification.

X. Acknowledgments

The work was done while the first author was at PhD Internship in INRIA, Rhone-Alpes, France. The authors like to thank Aurelien Francillon and Daniele Perito of INRIA Rhone-Alpes for their help while doing the work.

References

[1] Crossbow Technology Inc. Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[2] Datasheet of Sensirion SHT11: Humidity and Temperature Sensor on Telosb mote. Available at http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf.

[3] IEEE Wireless medium access control and physical layer specifications for low-rate wireless personal area networks. IEEE Standard, 802.15.4-2003.

[4] Imote2 Datasheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[5] IRIS Datasheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf.

[6] MICAz: Wireless measurement system. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[7] MTS-MDA Sensor Board Users Manual. Revision A, June 2007. PN: 7430-0020-05. http://www.xbow.com/Support/Support_pdf_files/MTS-MDA_Series_Users_Manual.pdf.

[8] NCDC: Global Measured Extremes of Temperature and Precipitation. Available at http://www.ncdc.noaa.gov/oa/climate/globalextremes.html#hightemp.

[9] OCXOs: Oven-Controlled Crystal Oscillators . http://www.wenzel.com/documents/ocxo.html.

[10] Specification of Sensirion SHT11: Humidity and Temperature Sensor on Telosb mote. Available at http://www.parallax.com/dl/docs/prod/acc/SensirionDocs.pdf.

[11] TCXOs: Temperature Compensated Crystal Oscillators. http://www.wenzel.com/documents/tcxo.html.

[12] TelosB mote platform. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[13] TinyOS: An open-source OS for the networked sensor regime. http://www.tinyos.net/.

[14] MPR-MIB Sensorboards Users Manual. Revision A. PN:7430-0021-08. http://www.xbow.com/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf, June 2007.

[15] M. E. Dyer. Linear time algorithms for two- and three variable linear programs. In *SIAM Journal on Computing*, volume 13, pages 31–45, 1983.

[16] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Programming Language Design and Implementation (PLDI)*, June 2003.

[17] A. Gehani and S. Chandra. PAST : Probabilistic Authentication of Sensor Timestamps. In *Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, FL, USA, 2006.

[18] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russel. Device identification via analog signal fingerprinting: A matched filter approach. In *The 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[19] J. Gupchup, R. Musaloiu-E., A. Szalay, and A. Terzis. Sundial: Using Sunlight to Reconstruct Global Timestamps. In *European Conference on Wireless Sensor Networks (EWSN)*, Cork, Ireland, 2009.

[20] J. Hall, M. Barbeau, and E. Kranakis. Detection of transient in radio frequency fingerprinting using signal phase. In *Wireless and Optical Communications*, 2003.

[21] 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 Globecom*, New Orleans, LO, USA, 2008.

[22] T. Kohno, A. Broido, and kc claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security & Privacy*, pages 211–225, May 2005.

[23] P. Levis. TinyOS Programming. Available at http://csl.stanford.edu/∼pal/pubs/tinyos-programming-1-0.pdf, June 2006.

[24] R. Maheshwari, J. Gao, and S. R. Das. Detecting Wormhole Attacks in Wireless Networks using Connectivity Information. In *IEEE Infocom*, Alaska, USA, 2007.

[25] M. Martinec. Temperature dependency of a quartz oscillator. http://www.ijs.si/time/#temp-dependency.

[26] N. Megiddo. Linear-time algorithms for linear programming in r^3 and related problems. In *SIAM Journal on Computing*, volume 12, pages 759–776, November 1983.

[27] D. Mills. RFC1305: Network Time Protocol (Version 3) Specification, Implementation. RFC1305, 1992.

[28] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *IEEE Infocom*, New York, Mar 1999.

[29] S. J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *ACM CCS*, pages 27–36, Alexandria, VA, US, October 2006.

[30] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil attack in sensor networks: analysis & defenses. In *Information Processing in Sensor Networks (IPSN)*, Berkeley, California, USA, 2004.

[31] V. Paxson. On calibrating measurements of packet transit times. In *SIGMETRICS' 98*, Madison, Wisconsin, USA, June 1998.

[32] K. B. Rasmussen and S. Capkun. Implications of Radio Fingerprinting on the Security of Sensor Networks. In *International Conference on Security and Privacy in Communication Networks (SecureComm)*, Nice, France, September 2007.

[33] C. Sharp, M. Turon, and D. Gay. TinyOS Extension Proposal(TEP) 102: Timers. Available at http://tinyos.cvs.sourceforge.net/*checkout*/tinyos/tinyos-2.x/doc/html/tep102.html.

[34] J. Wright. Detecting Wireless LAN MAC Address Spoofing. Available at http://www.net-security.org/dl/articles/wlan-mac-spoof.pdf.

[35] S. Zander and S. J. Murdoch. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *17th USENIX Security Symposium*, San Jose, CA, USA, 28 July-01 August 2008.

Digital Object Identifier: 10.4108/ICST.WICON2010.8503
http://dx.doi.org/10.4108/ICST.WICON2010.8503