



Berkeley  
UNIVERSITY OF CALIFORNIA

TROST  
Team for Research in Ubiquitous Secure Technology  
SUPERB-IT 2006

# Time Synchronization Attacks and Countermeasures in Sensor Networks

Jocelyn Adams, Brown University,  
Jocelyn\_Adams@Brown.edu



## Motivation

**Ubiquitous Computing:** the wireless sensor network (WSN)

- Low-power micro electro-mechanical communicating computing systems are being integrated with the environment
- Low-power computing systems characteristically use **serial communication**:
  - Requires extremely precise **time synchronization**
- Successful data transmission is dependent on time synchronization
  - Time synchronization protocol must be **secure**



# Time Synchronization of WSN

(Wireless Sensor Networks)

- Time of each node (without FTSP) is based on the natural frequency of the node's crystal clock
- Maximize time synchronization precision
- Minimize power consumption

**Clock drift:** compensated for by a time sync protocol

Slight change in crystal's natural frequency due to:

- Quality of crystal
- Environmental factors such as temperature

**Security has not yet been considered**



## Task

**Evaluate damage done to time sync of a WSN in which the FTSP is implemented by injecting false globalTime messages**

**Software:** Operating System: TinyOS

**Hardware:** Telos and TelosB motes, shown on the right

### Method

1. Set up multi-hop testbed
  - Motes must have TestTimeSync application installed
2. Program node to inject false globalTime
3. Use parsing program to read time stamps on incoming packets
4. Compare time synchronization errors of compromised network to uncompromised network



- Synchronizes time of a sender to possibly several receivers using a single radio message time stamped at both ends
- Makes use of dynamic root allocation in multi-hop networks
- Average of 1.6 $\mu$ s per hop (includes root failure)
  - measured by Akos Ledeczi at Vanderbilt University
- **Skew** Relative difference: frequency of global clock and local clock
  - Calculated using a linear regression based on least squares
  - Skew is very close to 1 and for precision is stored and referred to as (skew- 1)

## Conversion from localTime to globalTime :

$$\text{globalTime} = \text{localTime} + \text{offsetAverage} + \text{skew} * (\text{localTime} - \text{localAverage})$$

- localAverage: average of the of the most recently stored localTimes
- offsetAverage: average difference between localTime and globalTime

## localTime of the receiver in terms of the sender's calculated globalTime:

$$\text{localTime} = \text{globalTime} - \text{offsetAverage} + \text{skew} * (\text{globalTime} - \text{offsetAverage} - \text{localAverage})$$

- **( globalTime - offsetAverage ) is substituted for the second instance of localTime from the above calculation of globalTime -**
- **Assumes that globalTime and localTime are comparable**
- **Compromising the received globalTime works with this assumption to more efficiently compromise the localTime of the receiving node**

# Parsing Packets



- Packets are sent via serial communication
- The data is stored as a set of hexadecimal numbers

Standard Header									TimeSyncMsg																						
length	fcf	dsn	destpan	addr	type	grp ID	mote ID	Msg ID	globalTime				localTime				skew				synced	root ID		seq #	# entries						
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
16	01	08	2F	FF	FF	FF	FF	B0	34	06	00	D8	3	F5	16	FE	A1	EE	E4	40	00	A2	AF	ED	B5	01	0	05	0	85	04

## Decoding the globalTime and localTime

Both local and globalTime are 32 bits / 4 bytes

For the packet:

9B E4 FB 07

Time in seconds =

$$\frac{(\text{hex2dec}(9B) + \text{hex2dec}(E4)*2^8 + \text{hex2dec}(FB)*2^{16} + \text{hex2dec}(07)*2^{24})}{32678}$$

# Progress

## **Compilation of TestTimeSync**

- Resolved incompatibilities of TestTimeSync with Telos platform
- Changed code to inject compromised TimeSync messages

## **Small Simulation for Development of Parsing Application**

- 1 mote - TestTimeSync
- 1 mote - root – connected to PC via serial port
- 1 mote - base station – collects data to be parsed and read through an interface on the PC screen

## **Development of Basic Parsing Application**

Outputs:

- globalTime, localTime and real time

# Future Work

- Set up multi-hop network with the compromised TestTimeSync application running on one mote
- Parse input with developed parsing application
- Observe and evaluate offset caused by compromised
  
- Could later compromise motes in other ways and similarly evaluate the results