A critical look at sensor network security

A personal odyssey

Naveen Sastry (nks@cs.berkeley.edu)
November 17, 2005
Outline

1. Claim: conventional wisdom
2. Counter-claim: my view
3. Tools
4. Design example
5. The real worry
6. Recap & rant
I. A Claim
Conventional wisdom

Sensor network security is different from fixed infrastructure security
Conventional wisdom: evidence (1)

- Resource constraints
  - TinyPackets
  - TinyProcessors
  - TinyMemory
  - TinyOperatingSystems

Software solutions not feasible
e.g. no public key
Conventional wisdom: evidence (2)

- Mismatch between attacker & victim network
- No physical security
  (maybe the blackberries will bring some bears to watch over…)
- Compromised nodes
- Jamming
Hold up: What are the problems?

- Securing communications
  - Confidentiality
  - Integrity
  - Access Control
- Keying
  - Key distribution & update
  - Any-to-any communication
- Detecting compromised nodes
- Secure infrastructure services
  - Secure + { Routing, Localization, Time synchronization }
2. Counterclaim
Counterclaim

Sensor network security is different from fixed infrastructure security

Sensor network security is similar enough to fixed infrastructure security
Threat models

• Commercial (buildings/industrial plants/…):
  • Nodes under single administrative control
  • Minimal / low mobility
  • Single install time
  • No DoS worries
  • Pretty good physical security

• Millitary
  • Mobility!
  • Smart adversaries
  • Rich adversaries
  • DoS is the objective
3. Tools
Link layer encryption

SPINS (‘01)
  Sender, receiver synch problems
TinySec (’04)
  All software, <8% overhead
802.15.4 (’04)
  In hardware, essentially free

Secure 2-way communication
Assumes:
  Pre-shared keys
Prevents packet
  injection
  modification
  eavesdropping

• Based on symmetric key cryptography
  • Efficient (worst problem: ~8-16 bytes per message)
  • Shared keys required
  • Keys must be protected
Public key encryption

- Sizzle from Sun
- Uses elliptic curve cryptography
  - RSA is slow, large (1024 bit operations)
  - ECC is just as secure at 160 bits, much faster

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time* (s)</th>
<th>Data bytes</th>
<th>Code bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC secp160r1</td>
<td>0.81</td>
<td>282</td>
<td>3682</td>
</tr>
<tr>
<td>ECC secp224r1</td>
<td>2.19</td>
<td>422</td>
<td>4812</td>
</tr>
<tr>
<td>RSA 1024 (pub**)</td>
<td>0.43</td>
<td>542</td>
<td>1073</td>
</tr>
<tr>
<td>RSA 1024 (priv)</td>
<td>10.99</td>
<td>930</td>
<td>6292</td>
</tr>
<tr>
<td>RSA-2048 (pub**)</td>
<td>1.94</td>
<td>1332</td>
<td>2854</td>
</tr>
<tr>
<td>RSA-2048 (priv)</td>
<td>83.26</td>
<td>1853</td>
<td>7736</td>
</tr>
</tbody>
</table>

From Vipul Gupta, CENTS Retreat Jan 2005; CHES 2004
8 Mhz Atmel 128
Tamper resistance

- Single chips
  - Good also for security
- Careful hardware design
  - Eliminate side channels (power & timing attacks)
- Packaging
  - iButton & smartcards
  - ~ $1

Increasing cost
For the paranoid…

- **IBM 4758: No known physical attacks**
  - Mitigate cost: two tiered network
    - Trusted & protected infrastructure
    - Ordinary nodes

- **Jamming proof radios:**
  - Frequency hop based on shared secrets
  - Spread spectrum
4. Design Example
Securing refinery infrastructure [Pister TRUST]

- Need to be able to deploy additional nodes to replace busted ones
- Problem: How to get existing nodes to recognize new node? How to exchange keys?
Details...

- New node needs some credentials for master to accept it
- Standard options:
  - Key rotations
  - Public key
  - Location limited channel: bring new node next to master
- Alternative: PDA
5. The real worry
Wormholes: routing

- Forwards traffic
- No keying required
- Increases load
- Traffic analysis
- Selective forwarding
- Disrupts routing properties
Other wormhole attacks: localization

- Rebroadcasts at different signal strength
- Still no key required
Other wormhole attacks: time synchronization

- Delays traffic
- Still no key required
**Wormhole directions?**

- **Packet leashes:**
  - Nodes know layout
  - Have tight time synchronization (e.g. from GPS)
  - Time each packet in flight.
  - Doesn’t help for time synchronization application

- **Frequency hopping radios**
  - Must use keyed hop schedule
  - Must hop quickly (every symbol?)
  - Generally, military grade radios

- **Nothing cheap or particularly effective**