Leveraging Channel Diversity for Key Establishment in Wireless Sensor Networks

Matthew J. Miller
Nitin H. Vaidya
University of Illinois at Urbana-Champaign

April 27, 2006
The Promises of Sensor Networks

“Every sweet has its sour”
-Ralph Waldo Emerson

<table>
<thead>
<tr>
<th>The Sweet</th>
<th>The Sour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless links for easy, quick deployment</td>
<td>Tapping the channel is easier</td>
</tr>
<tr>
<td>Cheap and numerous devices</td>
<td>Difficult to avoid physical compromise</td>
</tr>
<tr>
<td>Small and energy-efficient devices</td>
<td>Resource constraints on cryptography</td>
</tr>
</tbody>
</table>
How Key Distribution Fits In

- Tapping the channel
  - Keys give confidentiality against eavesdropping
  - Keys avoid unauthenticated data injection
- Physical compromise
  - Distribution should be resilient to node compromise
- Resource constraints
  - Use symmetric key cryptography as much as possible
Problem Statement

- After deployment, a sensor needs to establish **pairwise symmetric keys** with neighbors for confidential and authenticated communication

- Applications
  - Secure aggregation
  - Exchanging hash chain commitments (e.g., for authenticated broadcast)
Design Space

- Every node deployed with global key
  - 😊 Minimal memory usage, incremental deployment is trivial
  - 🙁 If one node is compromised, then all links are compromised

- Separate key for each node pair
  - 😊 One compromised node does not affect the security of any other links
  - 🙁 Required node storage scales linearly with network size
Related Work

- Each sensor shares a secret key with a trusted device ($T$) [Perrig02Winet]
  - $T$ used as intermediary for key establishment
  - $T$ must be online and may become bottleneck

- Key Predistribution [Eschenauer02CCS]
  - Sensors pre-loaded with subset of keys from a global key pool
  - Tradeoff in connectivity and resilience to node compromise
  - Each node compromise reduces security of the global key pool
Related Work

- Transitory key [Zhu03CCS]
  - Sensors use global key to establish pairwise key and then delete global key
  - Node compromise prior to deletion could compromise entire network

- Using public keys (e.g., Diffie-Hellman)
  - High computation cost
  - But, is it worth it when this cost is amortized over the lifetime of a long-lived sensor network?
Related Work

- Broadcast plaintext keys [Anderson04ICNP]
  - If an eavesdropper is not within range of both communicating sensors, then the key is secure
  - Assumes very small number of eavesdroppers
  - No way to improve link security if eavesdroppers are in range
  - We propose using the underlying wireless channel diversity to greatly improve this solution domain
High Level View of Our Work

Bob

Channel 1

Channel 2

Alice

Eve
High Level View of Our Work

- Given $c$ channels:
  \[
  \Pr(\text{Eve hears Bob’s packet} \mid \text{Alice hears Bob’s packet}) = \frac{1}{c}
  \]

- If Alice hears $M$ of Bob’s packets, then the probability that Eve heard all of those packets is $\left(\frac{1}{c}\right)^M$

- As $\left(\frac{1}{c}\right)^M \to 0$:
  The packets Alice heard can be combined to create Alice and Bob’s secret key
Threat Model

- Adversary’s primary objective is to learn pairwise keys
  - Can compromise node and learn its known keys
  - Can overhear broadcast keys
- Adversary’s radio capability is similar to that of sensors [Anderson04ICNP]
  - Receive sensitivity
  - One radio
- Multiple adversary devices may collude in their knowledge of overheard keys
  - Collusion in coordination of channel listening is future work
- Denial-of-Service is beyond the scope of our work
Protocol Overview

- Predeployment
  - Give each sensor a unique set of authenticatable keys

- Initialization
  - Broadcast keys to neighbors using channel diversity

- Key Discovery
  - Find a common set of keys shared with a neighbor

- Key Establishment
  - Use this set to make a pairwise key that is secret with high probability
Phase 1: Predeployment

- Each sensor is given $\lambda$ keys by a trusted entity
  - Keys are unique to sensor and *not* part of global pool
  - $\lambda$ presents a tradeoff between overhead and security

- The trusted entity also loads the Merkle tree hashes needed to authenticate a sensor’s keys
  - $O(\lg N)$ hashes using Bloom filter authentication
  - $O(\lg \lambda N)$ hashes using direct key authentication
Phase 2: Initialization

- Each sensor follows two unique non-deterministic schedules:
  - When to switch channels
    - Chosen uniformly at random among $c$ channels
  - When to broadcast each of its $\lambda$ keys

- Thus, each of a sensor’s $\lambda$ keys is overheard by $1/c$ neighbors on average
  - Different subsets of neighbors overhear each key

- Sensors store every overheard key
Initialization Example

Nodes that know all of A and B’s keys:
- C, D, E
- C, E
- E
- Ø

= Channel 1
= Channel 2
Phase 3: Key Discovery

- **Goal**: Discover a subset of stored keys known to each neighbor

- All sensors switch to common channel and broadcast Bloom filter with $\beta$ of their stored keys
  - Bloom filter for reduced communication overhead

- Sensors keep track of the subset of keys that they believe they share with each neighbor
  - May be wrong due to Bloom filter false positives
Key Discovery Example

A’s Known Keys

B’s Known Keys

A and B’s Shared Keys

C’s Known Keys

A and C’s Shared Keys
Phase 4: Key Establishment

u’s believed set of shared keys with v = \{k_1, k_2, k_3\}

1. Generate link key:
   \[ k_{uv} = \text{hash}(k_1 \parallel k_2 \parallel k_3) \]

2. Generate Bloom filter for \( k_{uv} \):
   \[ BF(k_{uv}) \]

3. Encrypt random nonce (RN) with \( k_{uv} \):
   \[ E(RN, k_{uv}) \]

1. Find keys in \( BF(k_{uv}) \)
2. Use keys from Step 1 to generate \( k_{uv} \)
3. Decrypt \( E(RN, k_{uv}) \)
4. Generate \( E(RN+1, k_{uv}) \)
Simulation Setup

- Use *ns-2* simulator
- 50 nodes
- Density of 10 expected one hop neighbors
- By default, 15 nodes are adversaries and collude in their key knowledge
- By default, $\lambda$ is 100 keys/sensor
Results: The Advantage of Channel Diversity

Just one extra channel significantly improves security.
Results: Resilience to Compromise

Fraction of Links that are Secure

1.0

Fraction of Nodes that are Compromised

0.0

0.2

0.4

0.6

0.8

One Channel

Two Channels

≥ 3 Channels

Resilient to large amount of node compromise
Summary

- Key distribution is important for sensor networks
- Many distinct solutions have been proposed
  - No “one size fits all” approach emerges
- Our work is the first to propose using channel diversity for key distribution
  - Results show significant security gains when even *one* extra channel is used
Thank You!

http://www.crhc.uiuc.edu/~mjmille2
mjmille2@uiuc.edu
Wireless Channel Diversity

- Radios typically have multiple non-interfering, half-duplex channels
  - 802.11b: 3 channels
  - 802.11a: 12 channels
  - Zigbee (used on Telos motes): 16 channels

- At any given time, an interface can listen to at most one channel
Design Considerations

- Resource constrained
  - Energy, computation, memory, bitrate
- Large scale deployments
  - May need thousands (or more) of devices
- Topology may be uncontrolled
  - Specific device’s location unknown in advance
Using Path Diversity

- Path diversity can be used to get a small number of compromised links to zero
- Similar to multipath reinforcement proposed elsewhere
  - Node disjoint paths needed to combat node compromise
  - Only link disjoint paths needed to combat eavesdroppers

![Graph](image)

- Green = Secure Link
- Red = Compromised Link

$$k_{AD} = \text{hash}(k_1 || k_2)$$
Simulation Results for Example Topology

Number of Shared Neighbors Used

Fraction of Links That are Compromised

0.1

0.05

0.0

0

1

2

3

4

Number of Shared Neighbors Used
Merkle Tree Authentication

\[ C = \text{hash}(O_1) \]
\[ A = \text{hash}(C || D) \]
\[ R = \text{hash}(A || B) \]

Each sensor given \( R \) and \( O(\lg N) \) other hashes