SPINS: Security Protocol for Sensor Networks

Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar
Department of Electrical Engineering and Computer Sciences
University Of California, Berkeley

Why Security?
- Hostile environments
  - Battlefield sensing/actuation
- Safety-critical applications
  - Sensors in reactor complex
- Privacy intrusions
  - Employee tracking/monitoring
- Uncontrolled access

Attacks
- Class I
  - Clever or curious outsiders
- Class II
  - Knowledgeable insiders
- Class III
  - Funded organizations or governments

Threats
- Eavesdropping
- Message injection
- Message replay
- Message modification
- Impersonation
- Denial of service
- Malicious code
- Traffic analysis
- Side-channel analysis

Security Primitives
- Message confidentiality
- Authentication
- Service availability
- Message integrity
- Message freshness
- Non-repudiation
- Intrusion detection
- Intrusion tolerance or containment
- Audit trails
Requirements for Sensor N/W Security

- Data Confidentiality
  - From the observed communication pattern set up secure channels between nodes and base stations
- Data Authentication
  - Construct authenticated broadcast from symmetric primitives only
  - Introduce asymmetry with delayed key disclosure and one way function key chains
- Data Integrity
- Data Freshness
  - Recent data
  - No replay of data

Contributions

- The major contributions of this work are as follows
  - Exploring the challenges for security in the domain of sensor networks
  - Designing and developing µTESLA
  - Designing and developing SNEP
  - Designing and developing an authentication routing protocol using SPINS building blocks

Cryptographic Algorithms

- Asymmetric cryptographic
  - Both parties have public encryption and private decryption
  - Encryption data can be sent to an individual using a public key of that individual, decryption by individual using the public key
  - Digital signature proof of data sent by a particular individual
  - RSA, Digital Signatures
- Symmetric cryptographic
  - Both parties share a secret shared key
  - Secret key placed in both parties before issuance
  - DES, IDEA

Comparison

- Symmetric encryption requires less power and memory than RSA
  - 2K of memory (RSA encryption)
  - 8K of EEPROM (applications and OS)
- Security weakened in Symmetric encryption
- Asymmetric encryption is strengthen with the use of a CA
- The current cost of producing SPOMs which have the capacity to contain RSA is significant
- RSA may require a co-processor to do the calculations for encryption and decryption

The choice of these algorithms are need based

Problem Challenges and Constraints

- Energy consumption primarily
  - By radio communication
  - Need to minimize communication overhead
- Reliance on asymmetric digital signature
  - Long signatures with high communication overhead of 50-1000 bytes per packet
  - Very high overhead to create & verify signature

Security Building Blocks

Design a suite for Security Building blocks that are optimized

- Resource constrained environment
- Wireless communication
- Security Network Encryption Protocols (SNEP)
- Data confidentiality
- Data freshness
  - Strong freshness
  - Weak freshness
  - Total order on request-response pair
  - Used for time synchronization with the network
  - Partial message ordering
  - Sensor measurements
- Micro Timed Efficient Stream Loss-tolerant Authentication (µTESLA)
  - Authenticated broadcast
Sensor Hardware

- CPU: RISC-like, 8-bit, 4MHz
- Storage
  - 8KB Instruction flash
  - 512 bytes RAM
  - 512 bytes EEPROM
- Communication
  - 916MHz ISM band radio
- Bandwidth
  - 10Kbps
- OS
  - Tiny OS
- Available code space
  - 4500 bytes
  - OS code space is 3500 bytes

Tiny OS

- Simple component based OS
  - Subset of components used for application specific functionalities
  - Components are reentrant state machines
- Maintains high level of concurrency in a limited space
- Uses power efficiency
  - Spends unused CPU cycles in sleep
  - Turns radio off when not in use

http://web.cs.berkeley.edu/tos/

Communication Organization

Trust Setup

- Individual sensors are untrusted
- Goal: compromised sensor comprises that sensor only
- No trust on communication infrastructure
- Base station is trusted
  - Each node has a master key
  - This key is shared with the base station
- Each node trusts itself and its sensors
  - Nodes’ own clocks are trusted (by itself)

SNEP-mechanism

- Communicating parties share a counter, which is used as an Initialization Vector (IV)
- Counter is not sent with the message
- Block ciphers are in Counter Mode (CTR)
- Counter incremented after each block
- MAC used to achieve 2 party data authentication and data integrity
- Counter value is never repeated
- Counter value in MAC prevents replay attacks

SNEP-implementation

- Encrypted data: $E = \{D\} \oplus K_{encr, C}$
- MAC: $M = MAC(K_{mac, C}|E)$
- Keys derived from master key, $K$
- $A \rightarrow B: \{D\} \oplus MAC(K_{mac, C}|D) \oplus K_{encr, C}$

- $D$ is the data
- $C$ is the Initialization Vector (IV)
- $K_{encr}$ is the encryption key
- $K_{mac}$ is the MAC key
**Strong freshness in SNEP**

- Plain SNEP: only weak freshness
  - Sending order on messages within B only
  - No assurance to A that message created by B was in fact in response to event in A
- Use of nonce, \( N_i \), for strong freshness
  - \( A \rightarrow B: N_A, R_A \) (Requested Message)
  - \( B \rightarrow A: \{R_B\}_{MAC(K_{mac}, N_A | C)} \)
- If MAC is verified then A knows that B generated the response to its request

**SNEP-strong points**

- Low communication overhead
- Adds only 8 bytes per message
- Uses counter
- Counter value is kept at both end points
- Provides semantic security
  - Prevents eavesdroppers from interfering the message content from the encrypted message
- Data authentication, replay protection, and weak/strong message freshness

**TESLA vs. \( \mu \)TESLA**

- TESLA
  - Authenticates initial packet with a digital signature
  - Too expensive for sensor nodes
  - Disclosing a key in each packet requires too much energy (24 bytes/packet)
  - Expensive to store one-way key chain
- \( \mu \)TESLA
  - Uses symmetric mechanism
  - Discloses key once every epoch
  - Restricts number of authenticated senders

**\( \mu \)TESLA: An example**

- Base station (BS) broadcasts authenticated information to nodes
- BS and nodes are loosely time synchronized
- Each node knows the upper bound on max. synchronization error
- BS computes a MAC on the packet
- The key is secret at this point
- Sensor receives the packet & stores it in buffer
- BS broadcasts the verification key to all receivers
- Node verifies the authenticity of the key
- Node uses key to authenticate the packet in the buffer

**Key authentication by node**

- Each MAC key is a key of key chain
- It's generated by a publicly known one-way function, \( F \)
- Sender chooses the last key, \( K_n \), randomly
- Applies \( F \) to compute: \( K_0 = F(K_1) \)
- Each node performs time synchronization & to authenticate the key of the key chain

**Time synchronization**

- Assumptions
  - \( P_1 \) & \( P_2 \) sent in interval 1
  - \( P_3 \) sent in interval 2
  - \( P_4, P_5, \) & packet that discloses \( K_1 \) are lost
  - Keys are disclosed after 2 time intervals
- Interval 4: BS broadcasts \( K_2 \)
- Node authenticates by \( K_0 = F(F(K_2)) \)
- Also it knows that \( K_1 = F(K_2) \)
- So Packets \( P_1, P_2, \) & \( P_3 \) are authenticated
Implementation

- Block cipher
  - RC5
    - Small code size and high efficiency
- Encryption function
  - Same function for encryption and decryption
  - Stream cipher in nature
- Freshness
- Random-number generation
- Message authentication
- Key setup

RC5

- RC5

Created by Ron Rivest in 1995
- Main feature: data-dependent rotations
- Parameterized for word size, length of key, etc.
- Low memory requirements

Evaluation(1)

Code size breakdown for security modules

<table>
<thead>
<tr>
<th>Module</th>
<th>RAM size (bytes)</th>
<th>Version</th>
<th>Total Size</th>
<th>MAC</th>
<th>Encrypt</th>
<th>Key Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC5</td>
<td>80</td>
<td>Smallest</td>
<td>1594</td>
<td>480</td>
<td>382</td>
<td>622</td>
</tr>
<tr>
<td>Fastest</td>
<td>1826</td>
<td>Fastest</td>
<td>1826</td>
<td>506</td>
<td>508</td>
<td>622</td>
</tr>
<tr>
<td>Original</td>
<td>2674</td>
<td>Original</td>
<td>2674</td>
<td>1210</td>
<td>802</td>
<td>686</td>
</tr>
</tbody>
</table>

Evaluation(2)

Performance of Security primitives

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC (16 byte message)</td>
<td>600</td>
</tr>
<tr>
<td>Encrypt (16 byte message)</td>
<td>120</td>
</tr>
<tr>
<td>Key setup</td>
<td>8000</td>
</tr>
</tbody>
</table>

Applications

- Authenticated Routing
- Node-to-Node Key Agreement

Node-to-Node Key Agreement

- Symmetric protocol uses BS as trusted agent for key setup
- A & B don’t share secrets
- They trust a third party S, the base
- Secure key agreement + strong key freshness

\[
A \rightarrow B : N_A, A \\
B \rightarrow S : N_A, N_B, A, B, MAC(K_{BS}, N_A, N_B, A|B) \\
S \rightarrow A : \{SK_{AB}\}_{K_{AS}}, MAC(K_{AS}, N_A, B|\{SK_{AB}\}_{K_{AS}}) \\
S \rightarrow B : \{SK_{AB}\}_{K_{BS}}, MAC(K_{BS}, N_B, A|\{SK_{AB}\}_{K_{BS}})
\]
### Drawbacks of SPINS

- Information leakage due to covert channels
- Problem not addressed
- Does not deal with compromised sensors
- Does not deal with DoS attacks
- Hardware restrictions
  - Cannot provide Diffie-Hellman style key agreements
  - Cannot use Digital Signatures

### Conclusions

- Illustrates the utility of security building blocks
- Elements of design universal so can be applied to other sensor networks
- Communication cost small
- Elements of design influenced by available experimental platform