Introduction

- Wireless sensor network
  - Consists of a potentially large number of sensor nodes
  - A sensor node has limited battery power, memory, and computation capability.
  - Sensor nodes communicate over short distances through wireless links.
Motivation of Code Dissemination

- The need of removing bugs and adding new functionalities
- It is inefficient and sometimes impossible to physically access each node and update its program.

Code Dissemination

- The process of propagating a new code image to the nodes in an entire network

Deluge [Hui et al. SenSys’04]

- Most widely used
  - Completely implemented and included in recent TinyOS distributions
  - Targeted by all the existing secure code dissemination schemes

- Data representation
  
<table>
<thead>
<tr>
<th>Object</th>
<th>Page</th>
<th>Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4</td>
<td>0 1 2 3 4 5</td>
<td>3 4 5</td>
</tr>
<tr>
<td>s_x,</td>
<td>s_y,</td>
<td>s_z,</td>
</tr>
<tr>
<td>P</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
Deluge (Cont’d)

• Periodical advertisement
  – version of a code image
  – # of pages it has received for that version
  – dynamic adjustment of advertisement rate for energy efficiency
• Request based on Selective Negative Acknowledgment (SNACK)
  – a requested page number and a bit vector indicating the requested packets in that page.
  – Compute the union of the requested packets (via the bit vectors), and transmit the requested packets in a round-robin fashion.
• Page-by-page dissemination strategy
  – Spatial multiplexing
• Message suppression mechanisms for efficiency

Possible threats in Deluge

• Disseminating malicious code into a sensor network
  – Authentication of the packets composing a code image.

• Disrupting a normal code dissemination by forged request and advertisement packets
  – Authentication of request and advertisement packets

Existing Schemes for Secure Code Dissemination

• Sluice [Lanigan et al. ICDCS’06] and Berkeley approach [Dutta et al. IPSN’06]
  – A hash chain is constructed over pages (packets).
  – The hash image of the first page (packet) is signed.
  – Problems:
    • No immediate authentication of each code packet
    • No authentication of SNACK and advertisement packets
    • Possible DoS attacks by fake signature packets
Existing Schemes for Secure Code Dissemination (Cont’d)

- Colorado approach [Deng et al. IPSN’06]
  - Merkle hash tree to authenticate the packets in each page and hash chain over pages
  - Level-by-level transmission for the packets from a Merkle hash tree
  - Problems:
    - Large overhead (communication and dissemination delay)
    - Not fit into Deluge’s page-by-page transmission
    - Hash chain based method to mitigate DoS attacks by fake signature packets, but vulnerable to online attackers
    - No authentication of advertisements

Our Contribution: Seluge

- Seluge is a secure extension to Deluge.

- Seluge provides three layers of protection:
  - Immediate authentication of code dissemination packets
  - Authentication of page advertisement and SNACK packets
  - Anti-DoS protection for signature packets

DoS attacks exploiting authentication delays

- If a received packet cannot be immediately authenticated,
  - Buffering the packet
    - Consume memory and eventually exhaust resources
  - Dropping the packet
    - Inefficiency due to frequent retransmission
Immediate authentication of code dissemination packets:

- Deluge page-by-page dissemination strategy for immediate authentication

- Preparing a code image consists of the following two steps:
  - Preparing Code Dissemination Packets and Their Hash Images
  - Constructing Merkle Hash Tree and a Signature Packet

### Preparing Code Dissemination Packets and Their Hash Images

<table>
<thead>
<tr>
<th>Page 1</th>
<th>Pkt 1,1</th>
<th>Pkt 1,2</th>
<th>Pkt 1,47</th>
<th>Pkt 1,48</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 2</td>
<td>Pkt 2,1</td>
<td>Pkt 2,2</td>
<td>Pkt 2,47</td>
<td>Pkt 2,48</td>
</tr>
<tr>
<td>Page 3</td>
<td>Pkt 3,1</td>
<td>Pkt 3,2</td>
<td>Pkt 3,47</td>
<td>Pkt 3,48</td>
</tr>
<tr>
<td>Page 4</td>
<td>Pkt 4,1</td>
<td>Pkt 4,2</td>
<td>Pkt 4,47</td>
<td>Pkt 4,48</td>
</tr>
</tbody>
</table>

### Merkle Hash Tree

```
V_0,1 -- e_1
V_0,2 -- e_2
V_0,3 -- e_3
V_0,4 -- e_4
V_0,5 -- e_5
V_0,6 -- e_6
V_0,7 -- e_7
V_0,8 -- e_8
```

```
e_1-2
e_3-4
e_5-6
e_7-8
```

```
e_1-4
e_5-8
```

```
e_1-8
```
Transmission and Authentication of Code Dissemination Packets

• The base station first broadcasts the signature packet.

• Page-by-page transmission

• Immediate authentication of every code dissemination packets upon receipt
Authentication of the Signature Packet and the Packets in Page 0

\[ \{ V_{e_0, e_{e_0,e_0}, e_{e_0,e_0}} \} \]

Authentication of the Remaining Packets

\[ \{ V_{e_0, e_{e_0,e_0}, e_{e_0,e_0}} \} \]

Authentication of Page Advertisement and SNACK Packets

- \( \mu \)TESLA based approach
  - TinySeRSync [Sun et al. CCS’06]
  - Receiver and sender side delay
- Cluster key based approach
Cluster Key based Approach

- Cluster key setup between neighbors

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello message</td>
<td></td>
</tr>
<tr>
<td>Cluster key encrypted with a pairwise key</td>
<td></td>
</tr>
<tr>
<td>Cluster key encrypted with a pairwise key</td>
<td></td>
</tr>
</tbody>
</table>

Cluster Key based Approach (Cont’d)

- Outgoing page advertisement or SNACK packet
  - Authenticating the packet using a sender’s cluster key.
  - Including a unique sequence number (to prevent replay attacks)
- Incoming page advertisement or SNACK packet
  - Verifying its integrity using the sender’s cluster key

Cluster Key based Approach (Cont’d)

- Advantage
  - Simple and same degree of protection against external attackers as the µTESLA based approach
  - Immediate authentication of received packets
- Disadvantage
  - Unable to identify a compromised internal attacker
  - Local impact to the compromised nodes
Anti-DoS protection for signature packets

• Problem: by sending bogus signature packets, forcing nodes to perform expensive signature verifications

• Message specific puzzles
  – [Ning et al. TOSN’08]

Message Specific Puzzles

• Setup phase
  – Generating a one-way key chain
    \[ K_0 \rightarrow K_1 \rightarrow K_2 \rightarrow \cdots \rightarrow K_{n-1} \rightarrow K_n \]
  – Pre-distributing the key chain commitment \( K_0 \) to all sensor nodes before deployment
  – The puzzle key \( K_i \) is used for the \( i \)th version of the disseminated code image.

Message Specific Puzzles (Cont’d)

• Generating a message specific puzzle
  – \([M_i||\text{Sig}(i)||M_j]\) and \( K_i \) constitute a message specific puzzle.
  – Finding a valid solution \( P_i \)

\[ \begin{align*}
\text{H} &
\end{align*} \]

• Upon receiving a signature packet
  – Check the freshness of the puzzle key and then verify whether the puzzle key is valid or not.
  – The node verifies the puzzle solution.
Message Specific Puzzles (Cont’d)

- Easy verification by a regular sensor node, but hard to solve for an attacker
- An attacker cannot pre-compute puzzle solutions without the fresh puzzle keys.

Security Analysis

- Integrity of code images
  - Digital signature to authenticate the root of the Merkle hash tree in page 0
    - An adversary cannot compromise the base station.
  - Authentication of the packets in page 0
    - based on the security of Merkle hash tree
  - Authentication of the packets in page i+1
    - based on the one-way property of secure hash functions

Security Analysis (Cont’d)

- Resistance to DoS attacks exploiting:
  - Authentication delays
    - Page-by-page dissemination strategy
  - Expensive signature verifications
    - Message specific puzzles
  - Deluge propagation and suppression mechanisms
    - Cluster key based approach (only against external attackers)
Comparison with Previous Approaches

<table>
<thead>
<tr>
<th></th>
<th>Integrity of code images</th>
<th>Protection against DoS attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Immediate authentication</td>
<td>Authentication of adv.</td>
</tr>
<tr>
<td>Seluge</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Berkeley approach</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td>Colorado approach</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Seluge Security Properties

- Seluge guarantees the integrity of code images accepted at receiving nodes
- Seluge is resistant to all DoS attacks that manipulate code dissemination protocols
  - DoS attacks exploiting authentication delays
  - Each packet can be authenticated upon receipt
  - DoS attacks exploiting expensive signature verifications
  - Weak authentication of signature packets using MSP
  - DoS attacks exploiting Deluge propagation and suppression mechanisms
  - Authentication of Deluge maintenance packets

Seluge Implementation

- Implemented as an extension to Deluge 2.0
  - Java tools to construct and inject Seluge packets
  - nesC code for verification of Seluge packets
  - Use TinyECC 0.3 for signature operations
  - Use the hardware cryptography support in CC2420
- Will be released as an open source package

Table 1. Code size (bytes) on MicaZ.

<table>
<thead>
<tr>
<th></th>
<th>ROM</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seluge</td>
<td>22,226</td>
<td>1,123</td>
</tr>
<tr>
<td>TinyECC in Seluge</td>
<td>45,258</td>
<td>2,278</td>
</tr>
<tr>
<td>TinyECC</td>
<td>13,044</td>
<td>426</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Evaluated approaches
  - Seluge
  - Colorado approach [Deng et al. 2006]
    - We implemented it for comparison purposes
  - (Revised) Berkeley approach [Dutta et al. 2006]
    - Code provided by Prabal Dutta and David Culler
    - Revised to remove the DoS vulnerability
    - Does not include authentication of maintenance packets
  - Deluge
- Sluice [Lanigan et al. 2006] is not included
  - No way to fix Sluice

Experimental Evaluation (Cont’d)

- 65 MicaZ motes; 152.5 feet X 97 feet

Propagation Delay

The (revised) Berkeley approach doesn’t protect maintenance packets. When this is disabled in Seluge, the delay is reduced by 30 – 146 seconds.
Summary of Seluge

- Guarantee the integrity of disseminated code images
- Resistant to all DoS attacks that manipulate code dissemination protocol
  - Superior to all existing attempts
- Efficient
  - Least overhead among all existing attempts
- Will be released publicly as an open source package
Summary of Seluge

- Guarantee the integrity of disseminated code images
- Resistant to all DoS attacks that manipulate code dissemination protocol
  - Superior to all existing attempts
- Efficient
  - Least overhead among all existing attempts
- Will be released publicly as an open source package