CSC 774 Advanced Network Security


Slides by Sangwon Hyun

---

Outline

• Introduction
• Deluge overview
• Existing schemes for secure code dissemination
• Seluge
• Security Analysis
• Experimental evaluation
• Conclusion

---

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

- Object
  0 1 2 3 4 5
- Page
  0 1 2 3 4 5
- Packet
  3 4 5
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

Merkle Hash Tree
Constructing Merkle Hash Tree and a Signature Packet

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

Authentication of the Remaining Packets

Authentication of Page Advertisement and SNACK Packets

- µTESLA 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

```
Sender
Hello message
Cluster key encrypted
with a pairwise key
Cluster key encrypted
with a pairwise key
```

Receiver

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
    \[ K_0 \rightarrow K_1 \rightarrow K_2 \rightarrow \cdots \rightarrow K_{i-1} \rightarrow K_i \]
  – 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
  – \( i | M_j | \text{Sig}(i) | M_j \) and \( K_j \) constitute a message specific puzzle.
  – Finding a valid solution \( P_j \)

• 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>Sluice</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>
<tr>
<td>Seluge</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>
<thead>
<tr>
<th>Table 1. Code size (bytes) on MicaZ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
</tr>
<tr>
<td>Deluge</td>
</tr>
<tr>
<td>Seluge</td>
</tr>
<tr>
<td>TinyECC in Seluge</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 97feet

Propagation Delay

The (revised) Berkeley approach does not protect maintenance packets. When this is disabled in Seluge, the delay is reduced by 30 – 146 seconds.
Communication Overhead

![Graph showing communication overhead]

Propagation Dynamics

![Graph showing propagation dynamics]

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