Compromise-Resilient Anti-Jamming for Wireless Networks

X. Jiang, W. Hu, S. Zhu, and G. Cao
Department of Computer Science and Engineering
The Pennsylvania State University
Jamming Attacks and Existing Solutions

- **Jamming Attacks**

- **Current countermeasures**
  - PHY layer approaches – frequency hopping (UFHSS), spread spectrum (UDSSS), requires advanced devices.
  - Control channels – in a centralized way (need cluster head or trusted authorities)
  - Wireless Sensor networks – channel surfing
    - After jamming is detected, nodes switch to a different channel
      \[ C(n+1) = F_k(C(n)) \]
    - Cannot deal with insider attacks
Motivation

- Wireless sensor networks – resource constraint, distributed network, can be compromised.
- After an insider jamming is detected, how can the non-compromised nodes agree on a new group key in a fully distributed way?
- How do they distribute the new group key under the presence of the jammer.
Design Goal

• Scenario
  – Normal nodes are compromised and deceived as malicious inside jammers

• Goal
  – Construct and propagate a new group key to all non-compromised nodes under the presence of the compromised jammers
  – The new key is used to establish keyed secret channels which cannot be predicted by inside jammers and thus excludes compromised nodes from the network
System model

• Every pair of nodes shares a pairwise key
  – Many works can achieve that such as Blundo scheme
• Jammers have the same capabilities as the normal nodes
• Channel Switch Latency – the time to change from one channel to another.
  – In Mica2, the delay is about 34\text{ms}. In 802.11 Atheros chipset, the delay is 7.6\text{ms}. 
The Split-Pairing scheme

- Dealing with a single inside jammer
- Basic observations and scheme Overview
  - At any given time, the attacker can jam only one channel or neither of them when it is switching channel.
  - By splitting the network into two groups, the group free of jamming can propagate a new group key and all nodes in either group could get the key finally.
  - By pairing two groups, the new group key could be propagated to the nodes not obtaining the key.
The Split-Pairing scheme

- The scheme consists of three phases
  - Phase I – Channel Splitting
  - Phase II – Jamming and key propagation within one group
  - Phase III – Key propagation between groups
Phase II – Jamming and key propagation within one group

• The two group leaders propagate a new key within each subgroup.

\[ K = F(K_{1,\lfloor \frac{N}{2} \rfloor+1})(0) \]

• For example, group 1 broadcasts:

\[ M_1 = Mapping || E_{K_{1,2}}(T|2|K) || \ldots || E_{K_{1,\lfloor \frac{N}{2} \rfloor}}(T|\lfloor \frac{N}{2} \rfloor|K) \]

• Due to the channel switch delay of the jammer, each group has chances to finish the key propagation, e.g., for a key size of 8 bytes, three receivers only take \((3*8*8)/19.2Kbps = 10.4ms < 34 ms\)

• Based on the confirmations from each group member, there may be re-transmissions
Phase II – Jamming and key propagation within one group

- The optimal jamming strategy for a single jammer to increase the recovery delay is to actively jam two channels with equal probability.
  - Formal proof is not shown here

Consider the optimal jamming strategy and the case where each jam leads to a retransmission, the Phase II needs

\[ T \approx T_{kr} + T_{jr} = \frac{2(t_j + t_l)}{2(t_j + t_l) - \tau_0} \cdot T_{kr} \]

\( T_{kr} \) is the key propagation time without jamming and \( \tau_0 \) is the time for one round broadcast.
Phase III – Key propagation between groups

- After phase II, it is possible, one group finishes key propagation, but the other is not due to jammer.
- In phase III, we pair the two nodes with the lowest ids in two groups, then the second lowest and so on.
Phase III – Key propagation between groups

- The node without the key (e.g. node \( j \)) initiate key exchange protocol by sending
  \[ M_1 = T||j||MAC_{K_{i,j}}(T|j) \]
- Node \( i \) replies to \( j \) to propagate the group key \( K \)
  \[ M_2 = E_{K_{i,j}}(T|i|K) \]
- Last, node \( j \) returns a confirmation.
- If the attacker can guess out the communication channel, the pair can change to a new secret channel by \( C_{new} = H(K|1) \) and then \( C_{new} = H(K|2) \) if necessary.
Tree-based scheme

Motivation

- Suppose we have $m$ colluding jammers, $m$ channels could be simultaneously jammed.
- In split-pairing scheme, we need $m+1$ groups.
- Need more resource for new group key generation.
- Borrow the idea from the logical key tree construction, but do not assume the existence of secure channels between each pair of nodes. We construct such channels based on the pairwise keys.
Tree-based scheme

\[ l = 0 \]
\[ K_{1-8} = F(K_{15} | 0) \]

\[ l = 1 \]
\[ K_{1-4} = F(K_{13} | 0) \]
\[ K_{5-8} = F(K_{57} | 0) \]

\[ l = 2 \]
\[ K_{1-2} = K_{12} \]
\[ K_{3-4} = K_{34} \]
\[ K_{5-6} = K_{56} \]
\[ K_{7-8} = K_{78} \]

\[ l = 3 \]
\[ 1 \quad 2 \]
\[ 3 \quad 4 \]
\[ 5 \quad 6 \]
\[ 7 \quad 8 \]
An example

Round 1

1 \quad K_{1-4} = F(K_{13}|0) \quad \text{on } C_{12} = H(K_{12}|0)

2

3 \quad K_{1-4}

\quad \text{on } C_{34} = H(K_{34}|0)

4

5 \quad K_{5-8} = F(K_{57}|0) \quad \text{on } C_{56} = H(K_{56}|0)

6

7 \quad K_{5-8}

\quad \text{on } C_{78} = H(K_{78}|0)

8

Round 2

1 \quad K_{1-8} = F(K_{15}|0) \quad \text{on } C_{1-4} = H(K_{1-4}|0)

2 \quad 3 \quad 4

5 \quad K_{1-8} = F(K_{15}|0) \quad \text{on } C_{5-8} = H(K_{5-8}|0)

6 \quad 7 \quad 8
Tree-based scheme

- Key propagation protocol
- We apply similar reliable broadcast protocol
- For $k^{th}$ round, subgroup leader $2^k i+1$ propagates subgroup key to members $2^k i+2$ to $2^k (i+1)$ by

$$\text{Mapping}\||E_{K(2^k i+1)-(2^k (i+1))}(T\|K(2^k+1.i+1)-(2^k+1.(i+1)))$$

- Subgroup members replies as a conformation

$$E_{K(2^k i+1)-(2^k (i+1))}(T|i|K(2^k+1.i+1)-(2^k+1.(i+1)))$$

- Performance Analysis
- Computation Cost – $O(N*\log(N))C_H + O(N)C_F$
- Communication Cost – $O(N)C_{\text{broadcast}}$
- Storage Overhead – the same as split-pairing scheme
Sensor Testbed and Metrics

- 19 sensors at fixed locations in an indoor laboratory
- Each sensor has Chipcon CC1000 radio interface with 32 channels
- OS: TinyOS 2.0.1
- Implement channel switching by the interface `CC1000Control` and `tuneManual()` provided by module `CC1000ControlP`
- Implement jammer by disable CSMA
Channel Switching Latency

- Send 1-byte and switch to another channel at once
- Three switching policies: Random, increasing, decreasing
- Observations: 34ms for all three sensors with low variance due to digital frequency synthesizer.
The Impact of Jamming

Latency of the splitting phase (I and II) for a single group

Delay for both groups

![Graphs showing the impact of jamming on recovery time and latency for different node counts and jamming probabilities.](image-url)
The Performance of split-pairing

- a. Recovery latency of the splitting phase (Phase I & II) (jamming prob.=0.5, Size=16 nodes)
- b. Recovery latency of the splitting-pairing scheme (All 3 phases) (jamming prob.=0.5, Size=16 nodes) (Ts is the switch time from phase II to phase III)
Tree-based scheme

![Graph showing average recovery latency vs. jamming packet payload size for different numbers of jammers.](image1)

![Bar chart showing average recovery latency for different number of legitimate nodes.](image2)
Conclusions

• Proposed a split-pairing scheme to distribute a new group key under the presence of an insider jammer.
• Propose a tree-based scheme to deal with more jammers.
• Although the solution is presented on sensor networks, it can be extended to other ad hoc networks as long as there is a channel switch latency for the jammer.
Mitigating Routing Misbehavior in Disruption Tolerant Networks

Qinghua Li and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
Dropping Problem

- **DTN Routing**
  - Intermittent connectivity
  - Store-carry-forward
- **Routing misbehavior**
  - Drop packets when still having buffers
- **Consequences**
  - Fewer packets delivered to their destinations
  - Bandwidth wasted
Simulations on Effects of Misbehavior

*A transmission is wasted if the transmitted packet is dropped by misbehaving nodes before reaching the destination.*
New Challenges

• Intermittent and opportunistic connectivity
  – Existing dropping detection techniques for ad hoc networks not applicable
  – Neighborhood monitoring
    • Watchdog
    – Acknowledgement (ACK)
• Benign nodes may also frequently drop packets when buffer overflows
  – More obscure borderlines between benign nodes and misbehaving nodes if only dropping behavior is considered
  – Existing rating-based mitigation techniques cannot better differentiate these two
Overview of Our Approach

• Packet dropping detection
  – Contact record & record reporting
  – Misreporting problem: report forged records
    • Inconsistency between forged records
    • Each record flooded to some witness nodes
    • Witness collecting the appropriate records detects

• Mitigation of misbehavior
  – If misreport $\rightarrow$ blacklist
  – If not misreport $\rightarrow$ mitigation based on buffer size
Network & Security Models

• Network model
  – Dedicated buffer for packets received from other nodes
  – Loose time synchronization

• Adversary model
  – A few misbehaving nodes may collude
  – Colluders have out-band comm. channels

• Trust model
  – Public-key based authentication
  – Node ID forgery infeasible, i.e., no Sybil attack
  – E.g., Identity-based cryptography
Basic Approach

• Contact record
  – Node ID
  – Packets buffered before the contact
  – Packets sent/received during the contact
  – Sequence number
  – Timestamp
  – Signature

• Mandatory record reporting
  – A node reports its most recent contact records with $r$ (a system parameter) distinct nodes to the next contacted node
Misreport, Inconsistency, Detection – An Example

most recent record (forged)
Send: m2
Recv: none
Contact: M’
...

most recent record (forged)
Send: none
Recv: none
Contact: M
...

Send: none
Recv: none
Contact: M
...

Send: m2
Recv: none
Contact: M’
...

Flood (epidemic routing)

summary
Recv: Hash(None)

summary
Send: Hash(m2)

Detection!

X

W

Y
Dropping Detection

![Diagram of message flow and record tracking]

- **N1** sends **m1** to **M** at time **t1**
- **N2** receives **m1** at time **t2**
- **N2** sends message **m2** to **M**
- **N3** receives **m1** with the following record:
  - **Buffer:** m1
  - **Send:** m1
  - **Recv:** none
  - **Contact:** N1
  - **My Seq:** 10
  - **Time:** t1

- **N3** retransmits/replays **m1** at time **t3**

---

Each node and message transaction is represented with labeled nodes and arrows, and the records are detailed boxes at their respective times. The diagram illustrates the flow and the detection of dropped messages.
More on Misreporting

• Taxonomy
  – No collusion: replay old records
  – Collusion: forge record (previous example)

• Consistency Rules
  – A node uses a unique sequence number in each contact
  – For any two records signed by a node, the record that has a smaller timestamp also has a smaller sequence number
  – One unique record is generated in a contact
Mitigation

- Goal: limit packets sent to misbehaving nodes
- Misreporting node
  - Alarm flooded upon detection
  - Blacklisted
- Misbehaving nodes that honestly report
  - Hard to tell from benign nodes when real buffer size not pre-known
  - Heuristic: misbehaving nodes provide a very small buffer, if any, for the network
Observable Buffer Size (OBS)

- Given a number of observations, OBS is the largest observation s.t. $\Pr[\text{real buffer size} > \text{OBS}] \geq \sigma$
- Approach -- The Beta Probability Density Function
- Given $n'$ observations $B_0 < B_1 < \ldots < B_{n'-1}$

$$\text{RBM} = R_{\left[(n'+2)(1-\delta)-1\right]}$$

- Example: when $n'=38$ and $\sigma = 0.95$, OBS=$B_1$
  - 1 fault within 38 observations can be tolerated
- $\sigma$ tunes a tradeoff between mitigation efficacy and fault tolerance
Performance Evaluations

- MIT Reality trace
  - 97 nodes, 300 days, 110K contacts

- Compared schemes
  - No-Defense
  - Optimal
  - Our scheme

- Three parts
  - Performance of misbehavior mitigation
  - Effectiveness of misreporting detection
  - Cost
Performance of Mitigation

Misbehaving nodes drop all received packets.
Misbehaving nodes selectively drop received packets.
Misreporting Detection Rate and Delay

(a) Detection Rate

(b) Detection Delay
Flooding Attack

- Malicious nodes inject a large volume of packets into the network
  - Network congestion
  - Wasted resources: buffer, bandwidth, power
- DTNs especially vulnerable
  - Scarce contact opportunity
- Existing approaches
  - Rate limiting: filter excessive packets at egress router on the Internet or at neighbors in ad hoc networks
- How to enforce rate limiting in DTNs?
Rate Limiting to Deal with Flooding Attacks

- Problem: ensure no more than $L$ packets can be injected by a node within each time interval $T$
- A naive solution: collect-and-count
  - Difficult to collect all packets due to unreliable delivery in DTNs
  - High cost
- A new perspective: borrow ideas from packet dropping detection
Basic Idea

Detection

An example where $L=3$
Further research issues

• Witness selection methods
  – Random selection
  – Contact based selection
  – Social centrality based selection
  – Geography-based selection

• Record dissemination methods
  – Direct transmission
  – Multicast
  – Multi-path routing
Thank you ...

More information
http://mcn.cse.psu.edu