LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks

SENCUN ZHU
The Pennsylvania State University
and
SANJEEV SETIA and SUSHIL JAJODIA
George Mason University

Assumptions

- Sensors are not mobile
- Neighbor nodes are not known pre-deployment – Ariel scattering possible
- Sensors are deployed in a hostile environment
  - Adversary can eavesdrop, inject packets, and replay packets
  - Adversary can gain physical access to sensors
- Base station will not be compromised

Limited Resources

- Assumed to have 100s of Bytes to store keys
- Very limited power
- Example Implementation - Mica2 Motes
  - 7.8 Mhz processor
  - 4 KB RAM
  - 128 KB Non-volatile memory

Single Key Options

- Asymmetric Keys – Secure, but too processor and memory intensive to be practical.
- Shared Global Key – Most efficient solution, but least secure.
- Shared Pairwise Key – Most secure solution, but requires establishing and maintaining too many keys.

Design Goals

- Support Unicast, Multicast, and Broadcast Communications
- Support In-Network Processing (Data Aggregation and Passive Participation)
- Survivability
- Energy Efficiency
- Avoid Message Fragmentation

LEAP+ Overview

- Individual Key
- Pairwise Key
- Cluster Key
- Global Key

Individual Key

- Each node has a shared secret key with the base station
- Can be used to send private messages to and from base station
- Preloaded before deployment
- $IK_u = fK_m(u)$

Pairwise Key

- Shared synchronous key shared with each immediate neighbor
- Can't be preloaded since neighbor will not be known until after deployment
- Sensors can't be made tamper proof, but can be tamper resistant

Pairwise Key

- T_{min} - minimum time required for an adversary to compromise a sensor
- T_{est} – time required for a newly deployed node to discover its neighbors
- If $T_{est} < T_{min}$, then the newly deployed node can safely maintain more sensitive information for T_{est}

Key Pre-distribution

- Controller generates a key K_{IN}
- New sensors are loaded with K_{IN}
- Each node u can derive a master key

  $K_u = fK_{IN}(u)$

Neighbor Discovery

- New Sensor sends HELLO:

  u -> *: u.
- Existing neighbors send ACK:

  v -> u: v,MAC(K_v, u|v)
- u derives K_v using v and K_{IN}, then verifies the MAC.
- u and v generate their pairwise key

  $K_{uv} = fK_v(u)$

Key Erasure

- At time T_{min} it is no longer safe to know K_{IN}
- K_{IN} is deleted from sensor memory
- All K_v derived during neighbor discovery are deleted
- Sensor no longer has ability to discovery neighbors.

Extended Pairwise Key Scheme

- Provides some additional security if $T_{min} < T_{est}$ can't be guaranteed
- Nodes are loaded with K_{IN}^{i} for time T_i
- Node u must maintain all K_u^j for i<j<M
- Messages:
  - u -> *: u,i.
  - v -> u: v,MAC(K_v^i,u|v)
- Compromised node would only have valid K_{IN} for remaining T_i
- Costs more memory and requires more key erases

Cluster Keys

- Node u generates K_u^c and sends to each of its neighbors using the pairwise key
- Each neighbor v sends u its cluster key K_v^c
- If node u is revoked, each neighbor must create a new cluster key and transmit it to its remaining neighbors.

Global Keys

- Initially all nodes can be preloaded with global key
- Needs to be a simple and efficient method to distribute new global keys
- Sending new global key with each individual key is too expensive

Authenticated Node Revocation

- Controller uses µTesla to authenticate revocation message of u:

  Controller -> *: u, $fK_g^i(0)$,$MAC(k_i^T,u|fK_g^i(0))$
- After time interval, K_i^T is dispersed and message can be authenticated
- Neighboring nodes of u remove its pairwise key with u and distribute new cluster keys

Distribution of new global key

- Given that the network is organized as a breadth first spanning tree
- New global key is distributed down the tree using the cluster keys
- Distributing global key is expensive, but infrequent

Local Broadcast Authentication

- Cluster key is used for local broadcast
- To prevent Node impersonation, a one-way key chain is used
- Still weak security, but limits adversary to impersonating at most the number of packets previously sent out by the node.

Survivability

- The network needs to be able to survive when a small subset of nodes have been *unknowingly* compromised
- A compromised node *can* send bad readings to the base station and interfere with neighboring nodes
- A compromised node *cannot* impersonate the base station, another node, or have a large affect on non-neighboring nodes

Defense Against Routing Attacks

- A compromised node can spoof, alter, or suppress routing messages.
  - Not particularly effective because the sending node can detect the alteration and forward the dubious behavior to the base station
- Wormhole attack can be effective, but requires compromising two nodes – One close to base station, one in area of interest.

Performance: Node Revocation

- Symmetric operations for updating cluster keys:
  - Avg: $2\Sigma^{s}_{i=1}d_i/(N-1)$
  - Max: $\max(d_i) + s - 1$
- Max symmetric operations for distributing new global key: 2
- Given a network of size 1000, and connection degree of 20, avg computational cost is 2.7 symmetric operations.
- Communication costs are similar

Storage Requirements

- 1 individual key
- 1 global key
- d cluster keys, one for each neighbor
- L keys for one-chain for local broadcast
- Example: Given a key is 8 bytes, 20 neighbors, and a key chain of length 30, 736 bytes are required for storage.

Prototype Implementation

- Required 1.2 KB of RAM given 20 neighbors
- Took 8.5 seconds to discover all neighbors. This is less than the 10s of seconds previously shown a Mica2 Mote could be compromised, thus $T_{min} < T_{est}$ holds true.

Prototype Implementation

- ACKS were lost in neighbor discovery phase due to collisions.
- Solution: A three-way handshake was implemented to gain reliability. Handshake included passing cluster keys to not cause extra overhead.

Security Assessment

- Highly difficult to compromise entire network since obtaining K_{IN} is very unlikely
- Very resistant to cloning attacks
- Vulnerable to DNS attacks by sending many HELLO messages.
  - Three-way handshake developed during the prototype minimizes this risk
- Compromised nodes can cause localized damage

Open Issues

- Compromised node identification left undefined.
  - Could be implementation specific (known bad sensor readings, neighboring nodes observing alteration of messages)
- Can a compromised node claim a different node is compromised, thus convincing the base station from removing it from the network?
- How does updating keys handle situations of high packet loss similar to the ACK collision issue during node discovery?