Secure Sensor Network Routing: A Clean-Slate Approach

Bryan Parno, Mark Luk, Evan Gaustad, and Adrian Perrig

Carnegie Melon University

Overview

- Design a highly secure, highly available node-to-node sensor network routing protocol
- Security Goals
  - Prevention
  - Detection and Recovery
  - Resilience

Assumptions

- Existence of a Network Authority (NA) which provides each node with:
  - K_{NA}, ID_{α}, and $\{ID_{\alpha}\}_{K_{NA}^{-1}}$
  - A one way hash chain of challenges
- Reliable broadcast mechanism to communicate
- Very little movement amongst sensor nodes

Address and Routing Setup Overview

- Goals:
  - Provide a unique network address to each node that is based on network topology
  - Produce a routing table within each sensor nodes that provides an accurate path to every other node
  - Keep paths short and routing tables small

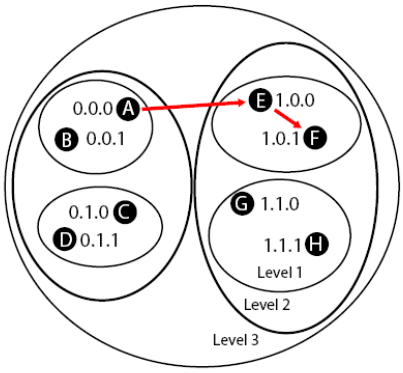
Initialization

- Each node is its own group
- Each node performs a neighbor discovery
  - Uses signed certificates to prevent adversaries from joining network
- Discovery phase is bound by time

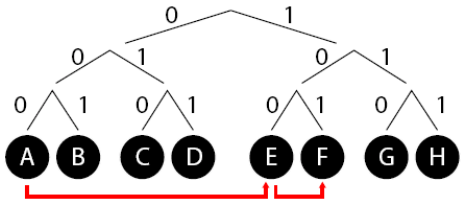
Recursive Grouping

- Group G requests to merge with its smallest neighbor G'
  - If G is G' smallest neighbor, G' accepts
- All nodes in G and G' compute their new group ID, size, and merge table
- Each node appends an additional bit to its address space to differentiate it from its new group members

Example Grouping Outcome



(a) **Recursive Grouping Outcome**



(b) **Network Addresses**

Prefix	Next hop
1.*	E (1.0.0)
0.1.*	C (0.1.0)
0.0.1	B (0.0.1)

(c) *A*'s **Routing Table**

Network Maintenance

- Node Death
  - Address space can remain unchanged
  - Node α requests alternative route from neighbors for every entry in routing table that includes the dead node
- Node Addition
  - Requires rerunning recursive grouping algorithm
  - Additions must be infrequent due to grouping algorithm overhead

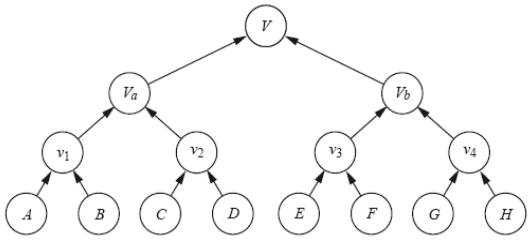
Packet Forwarding

- Packets received are forwarded using the routing table entry for the most significant address bit of the destination that differs from the current nodes address
- Example:
  - Destination Node D = 0.1.0
  - Current Node N = 0.0.1
  - Outcome: Node N sends the packet to the node in its routing table for prefix 0.1.*
  - Caveat: This next hop is not necessarily a member of group 0.1.*, but will always be a member of 0.*

Resilient Forwarding

- Multi-path forwarding to achieve high availability
- Amends the routing table to include three next hop options, {L,M,R}, to each group
  - Assumes a relatively dense network that results in receiving notifications of paths to another group from at least 3 neighbors
- Sender can produce a random direction string to attempt to avoid perceived problems
  - Example direction string = LMRLL
  - String must be as long as the number of physical hops to destination, which is unknown by the sender (only knows logical hops based on groups)
  - Random path may bypass problem node/region

Hash Trees



- $v1 = h(A||B)$, $Va = h(v_1||v_2)$, and $V = h(V_a||V_b)$
- VerifyLeaf: Sender, C, provides values C, D, v_2 and V_b so receiver can calculate: $V = h(h(v_1||h(C||D))||V_b)$
- VerifyTree: Uses a probabilistic approach to challenge C on the validity of its hash tree values

Grouping Verification Tree (GVT)

- Uses hash trees to prevent tampering with the recursive grouping algorithm
- Leaf values are node IDs
- Internal nodes are group IDs and calculated using hash function
- Each node maintains a merge table of groups it has merged with and their size

GVT Verification during a merge

1: G' announces its ID and size $(ID_{G'}, |G'|)$ to G
2: Group G chooses one of its nodes as a challenger C
3: C selects challenge C_k and broadcasts it to nodes in G
4: Nodes in G verify C_k is a correct challenge and edge nodes forward C_k to G'
5: Based on C_k, group G' chooses a responder node
6: Responder sends its certificate and merge table to G
7: Nodes in G perform the *VerifyTree* operation to authenticate the GVT for G'

GVT Continued

- After completion of the recursive grouping algorithm, all nodes know group id, V
- V can be used to authenticate any leaf using VerifyLeaf and that leaf's merge table
- Prevents Node replication and Sybil attacks

More Detection and Recovery

- Replication Detection Algorithm
  - Each node maintains a list of node IDs and addresses of its neighbors
  - Algorithm can detect a neighboring node claiming multiple addresses since it cannot fake its ID
- Honeybee Algorithm
  - If a legitimate node detects a malicious node, it broadcasts the implication
  - All other nodes immediately remove both the suspected malicious node and the detecting node from the network
  - Models a honeybee stinging an adversary, thus sacrificing itself
  - Prevents malicious nodes from removing more than 1 legitimate node from network
  - In a dense network, impact of losing 1 additional sensor node is minimal

Correctness Analysis

- Produces unique addresses to every node
- Produces deterministic routing between every pair of nodes
- Open issues:
  - How does the resilient forwarding handle sparse networks without 3 next hop options?
  - With the honeybee algorithm, how do nodes know which legitimate node to remove from the network when multiple nodes detect and implicate a malicious node?
  - How long do one-way hash chains last and is there a method for the NA to distribute new chains?
  - Protocol relies highly on reliable packet delivery

Performance Analysis

- Recursive grouping algorithm is logarithmic with respect to number of nodes
- Size of routing table, merge table, and address are logarithmic
- Open issues:
  - Performance impact of implementing reliable message delivery
  - Quantifying processing time and power consumption

Security Analysis

- Sybil nodes cannot enter the network due to the secure neighbor discovery process
- Malicious nodes can only successfully slander 1 node because of the honeybee technique
- Wormholes and other malicious nodes can be mitigated using the resilient routing technique
- Open issues:
  - A calculated slander attack with a small set of nodes could cause the network to become disjoint if all neighbors to a group were implicated

Simulation

- Compared to Beacon-Vector Routing (BVR) protocol
- Total Nodes: 100 and 500
- Nodes on average have 10 neighbors
- Tested with irregular topologies

Simulation Results

- BVR requires less setup overhead

	BVR		Proposed Protocol	
# Nodes	100	500	100	500
Avg Msg Sent	83	132	139	252
Max Msg Sent	111	201	199	392

- BVR requires more routing overhead due to necessary flooding when paths fail
- BVR is more adversely affected by voids in network topology

Implementation

- Used 16 Telos motes running TinyOS
- Code base was 21KB and routing data was 50 bytes
- Network successfully routed 100% of packets during a small test

Conclusions

- Only suitable for sensor networks that require all sensors to have the ability to communicate with all other sensors
- Protocol needs to handle unreliable packet delivery
  - Relies heavily on nodes assuming all nodes in group are performing the same operation
  - Relies on nodes assuming other nodes received its message/instruction
- Performance needs to be further tested.
  - Time and processor cycles required for setup
  - How much power is consumed during initialization and node additions?
  - Performance of removing nodes from network
- Further analysis
  - Can one-way key chains be re-established? What is the protocol?
  - Is the honeybee technique complete? What happens when multiple nodes accuse a single node?