Multigrade Security Monitoring for Ad-Hoc Wireless Networks

Matthew Tan Creti, Matthew Beaman□, Saurabh Bagchi, Zhiyuan Li□, Yung-Hsiang Lu

School of Electrical and Computer Engineering, Department of Computer Science(*) Purdue University

Mass
Macau SAR, P.R.C.
October 13, 2009

Motivation

- Ad-hoc multi-hop routing is used to span distances more than a single radio range
- Deployment in critical applications (e.g. infrastructure, monitoring, military) requires protection against sophisticated adversaries (e.g. colluding nodes)
- Three phases in data exchange between end-points
  - Route discovery <- the focus of this work
  - Data forwarding
  - Route repair
- Existing defenses fail against colluding nodes
  - Secure AODV (SAODV)
  - Local Monitoring
- Local monitoring is low cost (i.e. energy, traffic) because it is passive in the benign case
- Local monitoring suffers from high false alarms
- We are motivated to find defenses with low false alarms that are not expensive

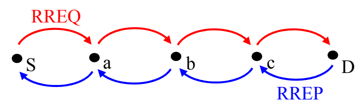
Contributions

- Multigrade framework combines local monitoring with more expensive defenses
  - Low cost
  - Low false alarms and low missed alarms
- Route Verification (RV) defends against colluding nodes
  - Route discovery
  - Route intrusion of data forwarding path
- Multigrade monitoring (MGM) protects route discovery
- Simulation results show
  - MGM only marginally more expensive than undefended route protocol
  - High coverage of attack detection

Outline

- Ad-Hoc On-Demand Distance Vector Routing (AODV)
- Known attacks
  - Route intrusion
  - Route discovery prevention
- Existing defenses
  - Secure AODV (SAODV)
  - Local Monitoring
- Route discovery attack definition
- Muligrade Monitoring framework
- Route Verification (RV)
- Multigrade Monitoring (MGM)
- Simulation results

Ad-Hoc On-Demand Route Discovery

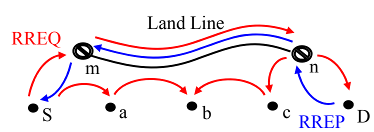


Routing tables at each node (destination:next hop):

	S:S	S:a	S:b	S:c
D:a	D:b	D:c	D:D	

- Source *S* wants to establish forwarding path to *D*
- *S* broadcasts route request (RREQ)
- Each node learns next hop to *S*
- *D* unicasts route reply (RREP) to *c*
- Each node in forwarding path learns next hop to *D*
- Forwarding path from *S* to *D* is established

Attack: Route Intrusion



Routing tables at each node (destination:next hop):

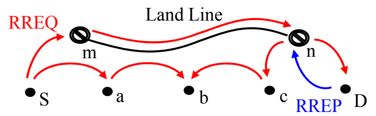
S:S S:a S:n S:n

D:m

= malicious node

- *S* broadcasts RREQ
- *m* forwards RREQ to *n* using out-of-band channel (e.g. land line)
- *n* repeats RREQ
- *D* unicasts RREP to *n*
- *n* forwards RREP to *m*
- *m* repeats RREP
- The *m*-*n* link is now part of the forwarding path, this can be used to selectively drop data packets
- **This paper is not focused on intrusion detection in a forwarding path. See X. Zhang, A. Jain, A. Perrig, "Packet-dropping adversary identification for data plane security," ACM CoNEXT, 2008.**

Attack: Route Discovery Disruption



Routing tables at each node (destination:next hop):

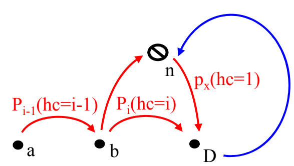
D:? S:S S:a S:n S:n

= malicious node

- *S* broadcasts RREQ
- *m* passes the RREQ to *n* using out-of-band channel (e.g. land line)
- *n* repeats RREQ
- *D* unicasts RREP to *n*
- *n* drops RREP
- *S* never learns forwarding path to *D*

Protection of Route Discovery: SAODV

Unprotected AODV



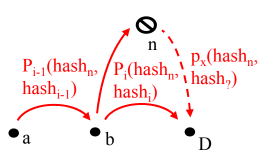
p_i(hc=i) = packet instance i with hop count i

RREQ

RREP

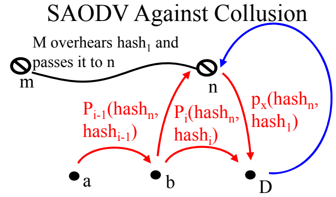
= malicious node

SAODV



hc = hop count
$hash_n = h^{n-x}(hash_x)$
where h is one way hash function and n is maximum hop count

SAODV Against Collusion

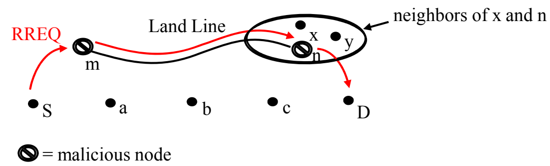


hc = hop count
$hash_n = h^{n-x}(hash_x)$
where h is one way hash function and n is maximum hop count

- Decreasing hop count allows malicious node to insert itself in route
- Secure AODV (SAODV) uses one way hashes to prevent malicious node from decreasing hop count
- SAODV is vulnerable to colluding nodes

M. G. Zapata, "Secure ad hoc on-demand distance vector (saodv) routing," September 2006, http://personals.ac.upc.edu/guerrero/papers/draft-guerrero-manet-saodv-06.txt

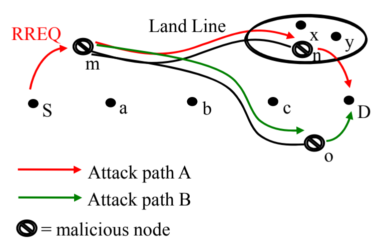
Protection of Route Discovery: Local Monitoring



- Assume *secure neighbor discovery* (all nodes know neighbors and neighbor's neighbors)
- Require any node forwarding a routing message to declare where it received the message from
- *n* cannot claim it received the RREQ from *m* because *D* knows that *m* and *n* are not neighbors and ignores the message
- If *n* claims it received the RREQ from a neighbor (e.g. *x*), then *y* (a neighbor of both *n* and *x*) will detect that *n* has fabricated a message
- **Problem (false alarms):** *x* might forward a message to *n*, but due to collision *y* fails to overhear the message, when *n* forwards the message to *D*, *y* now falsely detects n as malicious
- Because of false alarms a *threshold* is used, such that the rate of alarms must exceed some value before the monitor decides that a node is acting maliciously

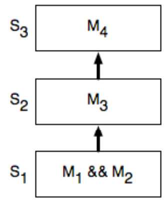
I. Khalil, S. Bagchi, and N. B. Shroff, "Liteworp: Detection and isolation of the wormhole attack in static multihop wireless networks," Comput. Netw., vol. 51,no. 13, pp. 3750–3772, 2007.

Colluding Nodes Defeat Local Monitoring



- Strategies to defeat local monitoring:
  - Malicious nodes stay just under the threshold for detection
  - Spread out the attack over multiple colluding nodes (e.g. attack path A and B)
- Using n attack paths the adversary can commit n times more malicious actions globally without any of the malicious nodes being detected locally
- Adversary can disrupt more routes discoveries without detection by increasing the number of colluding malicious nodes

Multi-grade Monitoring Framework

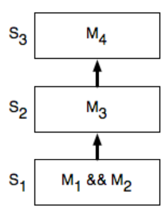


Highest cost, lowest false alarms

Lowest cost, highest false alarms

- Local monitoring is an example monitor with high *false alarms* (false positives) but low *missed alarms* (false negatives)
- Monitors with low false alarms are generally more costly on resources
- We can arrange monitors into stages, such that low cost monitors (e.g. local monitors) detect *necessary* conditions for an attack to succeed, and then trigger monitors that detect *sufficient* conditions for an attack to succeed (monitors with no false alarms)

Multi-grade Monitoring (Continued)



Highest cost, lowest false alarms

Lowest cost, highest false alarms

- Stages S_1 to S_n, where lower number stage triggers higher number stage (i.e. lower stages filter out events that are not malicious)
- Monitors can be combined in a stage to reduce rate for false alarm (e.g. M1&&M2)
- The final stage S_n triggers *diagnosis and attack mitigation*
- Observe the following points of the system:
  1. As long as every stage has no missed alarms then the system will have no missed alarms
  2. If stages are ordered by false alarm rate such that $FA(S_1) > \ldots > FA(S_n)$ then $FA(system) \leq FA(S_n)$.
  3. Combining points 1 and 2, if the missed alarm rate at all stages is zero and the false alarm rate S_n is also zero, then the system has perfect detection.
  4. Let the rate of monitored events be *r*. The cost to the system, in the benign case where no attack is present, is $r*C(S_1)+r*FA(S_1)*C(S_2)+\ldots+r*FA(S_1)*\ldots*FA(S_n)*C(S_n) << r*C(S_n)$

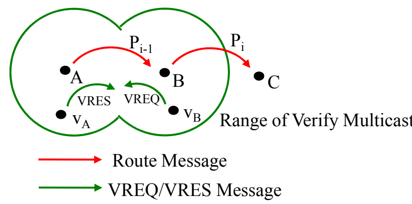
Route Discovery Attack Definition

- **Goal:** Adversary wants to prevent route discovery for a period of time
- Adversary may compromise nodes giving it "*insider*" malicious nodes
- Malicious nodes may *modify, fabricate*, *drop* routing packets
- Malicious nodes have no limit to *collusion*

Necessary Conditions for Route Discovery Disruption

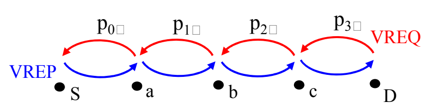
- One of these must occur for route discovery disruption:
  1. A node modifies a route packet to insert itself in the route (e.g. decreases hop count)
  2. A node fabricates a packet (e.g. forwarding a packet through a wormhole)
- Local monitoring detects packet modification with a low false alarm rate
- Local monitoring may have high false alarm detecting packet fabrication
- So we need a new defense mechanism for the highest stage that can detect fabrications with low false alarms

Route Verification (RV) Protocol



- When node *B* forwards a *packet instance* P_i from *A* to *C* a verifier node of *B* multicasts a *verify request (VREQ)*
- A *verifier node* of a node, is any neighbor of that node
- The *VREQ* is multicast to all neighbors of both *A* and *B*
- A *verify response (VRES)* informs that a packet instance has been correctly forwarded
- When the neighbors of *A* receive a *VRES* for packet instance P_{i-1} a verifier of *A* sends a *VRES* for packet instance P_i

What RV Provides

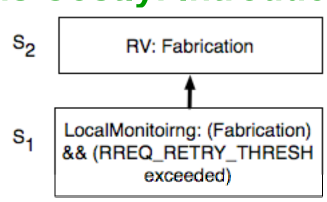


- The verification continues in a chain
- **Invariant 1:** If a packet instance p_i is marked *verified*, then the previous packet instance p_{i-1} has also been marked verified
- **Invariant 2:** The packet instance p_S is determined verified, only if node *S* broadcasts a *VRES* to its neighbors that the packet is correct
- **Lemma:** By induction, if a packet p_i is marked verified then none of the packet instances p_S to p_i have been *modified* or *fabricated*

How to Use RV

- Diagnosis of the *VREQ* and *VRES* messages can easily allow nodes to determine the source of the malicious actions when the nodes are a neighbor of the malicious node
- This allows the neighbors of the malicious node to isolate it from the network
- We assume the origin node repeats a RREQ after a timeout period
- Eventually, all of the malicious nodes will be isolated and the route will be established
- Therefore, we are protected against the route discovery disruption attack

But RV is Costly: Introducing MGM

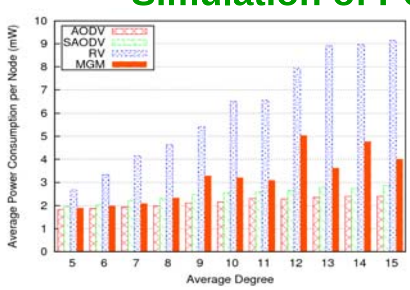


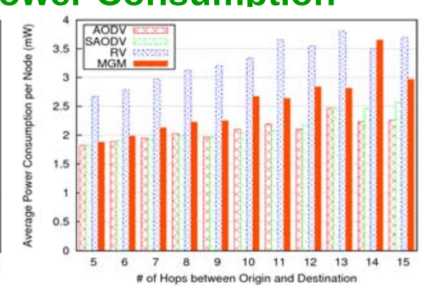
- Necessary conditions to prevent route establishment over multiple route requests:
  1. a packet instance is fabricated
  2. verifier node in the network hears a version of each repeated route request (it may be a fabricated or modified version)
- Local monitoring detects condition 1 with low missed alarms and condition 2 can be observed from the route layer with low missed alarm
- The sufficient condition is to repeat the packet modification or fabrication over multiple route requests
- The sufficient condition is detected by RV

Simulation Setup

- 200 nodes placed randomly in a square
- The size of the box is varied to get node topologies with different number of average degree
- Collisions are simulated in TinyOS 2.x radio model
- BMAC (low power listen) is simulated so that there is an energy advantage to reducing number of packets sent in network
- During the run of a simulation random source and destination node pairs try to establish routes
- No data packets are sent on the routes

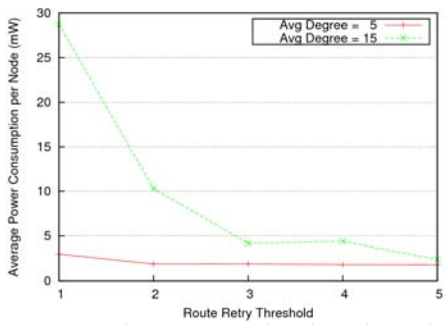
Simulation of Power Consumption





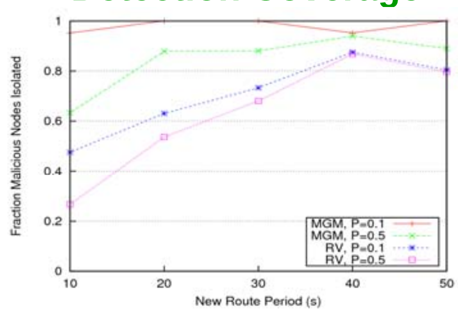
- RV and MGM are compared with AODV and SAODV (even though SAODV is not secure against wormhole attacks)
- RV quickly becomes more expensive than other protocols with increasing average degree and number of hops
- For low average degree and number of hops MGM is no more expensive than regular AODV

Tuning MGM



- The key parameter in MGM is the threshold number of route retries
- The parameter quickly stabilizes even for high average degree
- For our particular simulation 3 was the best choice

Detection Coverage



- We look at the effect of high traffic on isolation coverage
- New origin destination pairs attempt to establish routes every route period
- P is the probability of a route message being attacked by an adversary
- High traffic reduces isolation coverage, after a point traffic saturates the network
- MGM greatly reduces the amount of traffic when compared to RV, and therefore has better isolation coverage for all points

Conclusion

- Developed a low-cost technique for detecting attacks against route establishment
  - Structure the available monitors to achieve lower resource cost with negligible decrease in detection performance
  - Detect necessary conditions first with low cost monitors and then sufficient conditions with higher cost monitors
- Developed a technique – Route Verification (RV) – for detection of attacks with colluding malicious nodes
  - RV surpasses previous work in its ability to detect arbitrarily powerful collusion
  - RV has very low false alarm and missed alarm rates, but it is expensive in the additional traffic that it generates