On Identifying Malicious Nodes in Ad Hoc Networks

Xu Su, Rajendra V. Boppana
Department of Computer Science, The University of Texas at San Antonio
San Antonio, TX 78249
xsu@cs.utsa.edu, boppana@cs.utsa.edu

ABSTRACT

Route falsification attacks are easy to launch in mobile ad hoc networks with on demand routing protocols that employ network-wide flooding of control packets for route discoveries. To mitigate this attack, we propose a p-hop crosscheck mechanism that requires nodes p, $p \geq 2$, hops apart to authenticate and verify route reply packets using pair-wise shared keys. The crosscheck can detect route falsification by non-colluding malicious nodes on-the-fly; furthermore, it can identify a group of at most $p+1$ nodes that contain the malicious nodes that caused the route falsification. Unlike intrusion detection techniques, which require extensive monitoring and sampling, the proposed crosscheck mechanism is light-weight and fast. Therefore, the proposed crosscheck mechanism can be used to augment the existing secure routing protocols and improve intrusion detection capability. We implemented 2-*hop* crosscheck for AODV in the Glomosim simulator. Using simulations, we show that 2-*hop* crosscheck mitigates attacks by multiple malicious nodes with negligible performance impact.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security

Keywords

Ad Hoc Networks, Secure Routing Protocols

1. INTRODUCTION

A mobile ad hoc network (MANET) consists of several wireless hosts that are capable of communicating with each other without the use of a network infrastructure or any centralized administration. To facilitate multi-hop communication between non-neighbor nodes, other nodes must act as routers. Since MANETs can be set up easily and inexpensively, they have a wide range of applications, especially in military operations and emergency and disaster relief efforts. However, MANETs are more vulnerable to security attacks than conventional wired and wireless networks due to the open wireless medium used, dynamic topology, distributed and cooperative sharing of channels and other resources, and power and computation constraints. Of particular interest and challenging are the active route falsification and resource depletion attacks.

We are interested in preventive solutions to route falsification attacks in on demand routing protocols for MANETs. In a route falsification attack, a malicious node falsifies route requests and/or route reply packets to indicate a better (shorter or fresher) path to the source of a data connection, make disproportionately large portion of traffic go through them. When the source selects the falsified path, the malicious node drops data packets it receives silently (denoted, blackhole attack) or forwards the packets but keeps the information to conduct analysis of communication patterns such as sender-recipient matchings, traffic timing, volume, and shape [17].

The current solutions [2, 10, 14, 20, 18, 12] to mitigate route falsification have easily exploitable security holes (see Section 4) and often do not consider performance implications of the security mechanisms proposed. In this paper, we propose a p-*hop* crosscheck mechanism to identify malicious nodes falsifying route request and route reply control packets in on demand route discoveries. The proposed mechanism uses pair-wise symmetric keys to authenticate and verify control packets by nodes that are p hops apart. We show that this can detect route fabrication by non-colluding malicious nodes. Furthermore, it can identify a group of at most $p+1$ nodes that contains the malicious nodes. This mechanism can be used to secure both on-demand table-driven routing protocols and source routing protocols such as Ad hoc On-demand Distance Vector routing protocol (AODV) [15] and Dynamic Source Routing (DSR) [11]. Compared to the current intrusion detection techniques, the crosscheck mechanism is light-weight and fast. Therefore, the proposed crosscheck mechanism can be used to augment the existing secure routing protocols and improve intrusion detection capability. We implemented 2-*hop* crosscheck (a special case of p-hop crosscheck) on AODV. Our results indicate that 2-*hop* crosscheck mitigates route falsification attacks effectively.

The rest of the paper is organized as follows. Section 2 presents the p-*hop* crosscheck mechanism. Section 3 presents the performance of the proposed mechanism. Section 4 describes related results in literature. Section 5 concludes the paper.

2. CROSSCHECK MECHANISM

We begin by presenting the basic route discovery mechanism and maintenance used in existing on demand routing protocols. Most

This research was partially supported by NSF grants EIA-0117255, CRI- 0551501 and AIA grant F30602-02-1-0001.

IWCMC'07, August 12–16, 2007, Honolulu, Hawaii, USA.

of the on-demand routing protocols use route discovery to learn new routes and route error propagation to remove stale routes. The route discovery consists of two stages. (1) *Route request stage* – the source node floods the network with a route request control packet (RREQ), and each node (with the exception of the destination) rebroadcasts the RREQ the first time it hears. (2) *Route reply stage* – upon receiving a RREQ, the destination sends a route reply packet (RREP), which is propagated to the source in the reverse path of the RREQ.

We use the following attack model. Malicious nodes attempt to falsify route requests and route replies so that they are in disproportionately large number of routes and impact the network performance by dropping all data packets passing through them. Also, malicious nodes do not collude, an assumption commonly made in the design of secure communication protocols, and falsify only a small fraction of route requests they forward so that an intrusion detection system cannot easily identify them by examining their reply/request volume.

In p-*hop* crosscheck, nodes that are p-hops apart authenticate RREP messages in the route discovery. Each route request (RREQ) or route reply (RREP) control packet contains a node list which includes the last p-1 nodes in its path. So, upon receiving RREP, each node knows its neighbors within p hops away in both directions along the path from the source to the destination. Each intermediate node records the node list carried by a RREQ (or RREP) into its routing table.

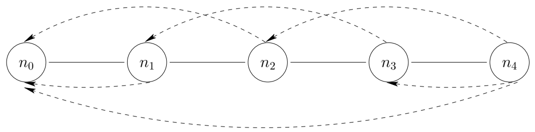

Figure 1: 2-hop crosscheck example: solid link indicates wireless link and dashed arc indicates node pairs that perform a crosscheck

We use a simple example to illustrate p-hop crosscheck mechanism shown in Figure 1. In this example, there is a 5-node path from source n_0 to destination n_4, and 2-*hop* crosscheck is used. Each node (e.g., n_3), called authentication node, creates a message authentication code (MAC) using the shared key between current node (e.g., n_3) and the node (e.g., n_1), called verification node, which is 2 hops away from it along the path to the source n_0. The verification node (e.g., n_1) will check if middle nodes (e.g., n_2) between the authentication node (e.g., n_3) and itself (e.g., n_1) falsify the RREP message by verifying the MAC value created by the authentication node (e.g.., n_3).

2.1 Route Discovery

We modify the basic route discovery to identify malicious nodes falsifying RREQs and RREPs. For now, let us assume that p-*hop* crosscheck does not allow intermediate node to originate a RREP even if it has a fresh route to the destination. Later, we will discuss the necessary changes to the cross-check mechanism to remove this restriction and improve network performance.

We use the following notation to describe the p-hop crosscheck mechanism.

- p: design parameter in crosscheck mechanism ($p \geq 2$).
- n_i: the i-th node from source to destination in a route discovery. The whole path can be represented as $n_0, n_1, n_2, \ldots, n_l$, where n_0 is source, n_l is destination, and l is total number of hops of this path; $p \leq l$ always holds.
- K_{n_i,n_j}: shared key between n_i and n_j; $K_{n_i,n_j} = K_{n_j,n_i}$.
- $MAC_{n_i,n_j}(X)$: message authentication code computed over message X using shared key K_{n_i,n_j}.
- $FH^p_{n_i}$ (forward hop of n_i): node which is p hops away from the node n_i along the path to n_l in a route discovery, it is given by

$$FH^p_{n_i} = \begin{cases} n_{i+p}, & i+p<l \\ n_l, & otherwise \end{cases} \tag{1}$$

- $BH^p_{n_i}$ (backward hop of n_i): the node which is p hop away from the node n_i along the reverse path to n_0 in a route discovery, it can be obtained by

$$BH^p_{n_i} = \begin{cases} n_{i-p}, & i-p>0 \\ n_0, & otherwise \end{cases} \tag{2}$$

- $FHL^p_{n_i}$ (forward hop list of n_i): intermediate nodes which are within $p-1$ hops away from the node n_i along the path to n_l in a route discovery, it is given by

$$FHL^p_{n_i} = \begin{cases} (), & i >= l-1 \\ (n_{i+1}, \ldots, n_{i+p-1}), & i+p-1<l \\ (n_{i+1}, \ldots, n_{l-1}), & otherwise \end{cases} \tag{3}$$

- $BHL^p_{n_i}$ (backward hop list of n_i): nodes which are within $p-1$ hops away from the node n_i along the reverse path to n_0 in a route discovery, it can be obtained by

$$BHL^p_{n_i} = \begin{cases} (), & i \leq 0 \\ (n_{i-p+1}, \ldots, n_{i-1}), & i-p+1>0 \\ (n_1, \ldots, n_{i-1}), & otherwise \end{cases} \tag{4}$$

- fh_{n_i} (forward hop count): the number of hops away from n_l; known to n_i after it receives a RREP.
- bh_{n_i} (backward hop count): the number of hops away from n_0; known to n_i after it receives a RREQ.
- $hop(n_i, n_j)$: number of hops which n_i is away from node n_j in a route discovery, this value is known to n_i.

2.1.1 Route Request Stage

When source n_0 needs to send data to destination n_l, it initiates a route discovery and broadcasts a RREQ, which contains the following information: message type ($rreq$), source id (n_0), destination id (n_l), source request number ($n_0\#$), backward hop count ($bh_{n_0} = 0$), and a MAC ($M_{n_0,n_l} = MAC_{n_0,n_l}(n_0\#)$). M_{n_0,n_l} is used for destination to verify the authenticity and freshness the RREQ generated. Each RREQ forwarded from intermediate node n_i contains a node list which includes all the nodes traversed within $p-1$ hops away from n_i back to n_0, denoted as $BHL^p_{n_i}$ given in (4). When an intermediate node n_i receives a RREQ from n_{i-1}, it will record its backward hop list ($BHL^p_{n_i}$) into its routing table and also update the backward hop list in the received RREQ before forwarding it.

2.1.2 Route Reply Stage

When a RREQ arrives at destination n_l, n_l will verify if this RREQ is originated from source n_0 by checking M_{n_0,n_l}. If the originality of the RREQ can be verified, n_l generates a RREP and sends it back to n_0 along the reverse path of RREQ received. The RREP sent or forwarded from node n_i contains both invariant (nonmutable) control information and variable (mutable) control information. The invariant part contains the following fields: message type ($rrep$), n_0, n_l, $n_0\#$, destination reply number ($n_l\#$), hop count (l), and a MAC, $M_{n_l,n_0} = MAC_{n_l,n_0}(rrep, n_0\#, n_l\#, l)$. The variant part, which may be changed by intermediate nodes, contains the following information: backward hop list $BHL^p_{n_i}$, current node id n_i if it is not destination, forward hop list $FHL^p_{n_i}$, number of hops away from destination fh_{n_i}, and a MAC list. The MAC list contains p MAC values:

$$M^p_{(FH^{p-1}_{n_i}, BH^1_{n_i})}, M^p_{(FH^{p-2}_{n_i}, BH^2_{n_i})}, \ldots, M^p_{(FH^0_{n_i}=n_i, BH^p_{n_i})},$$

where

$$M^p_{n_i,n_j} = MAC_{n_i,n_j}(rrep, n_0, n_l, n_0\#, n_l\#, l, \\ fh_{n_i} + hop(n_i, n_j), BHL^p_{n_i}) \qquad (5)$$

When an intermediate node n_i receives a RREP, it checks the MAC ($M^p_{FH^p_{n_i},n_i}$) created by node $FH^p_{n_i}$. If the MAC does not match, one or more nodes among the $p-1$ nodes between n_i and node $FH^p_{n_i}$ falsify the RREP. If malicious behavior is detected, n_i drops the RREP reports the falsification to the source, n_0. Otherwise it removes $M^p_{FH^p_{n_i},n_i}$ from the MAC list in the RREP and computes a new MAC ($M^p_{n_i,BH^p_{n_i}}$) and add it into the end of the MAC list and forward the RREP to its backward hop.

When the source receives a RREP, it will verify the originality of the RREP by checking M_{n_l,n_0}. It will also verify each MAC value in the MAC list contained in the received RREP. If all MAC values are verified successfully, the source will accept this RREP and update its route tables or route caches appropriately.

2.2 Special cases of p-*hop* crosscheck

We are interested in two special cases of *p-hop* crosscheck: 2-*hop* crosscheck (p = 2) and *complete* crosscheck (p = l). We will discuss these two cases below.

2.2.1 2-*hop* crosscheck

2-*hop* crosscheck can be used to secure AODV with slight modification. A route discovery example is given in Figure 2 and its related security property graph in route reply stage is shown in Figure 1.

$$\begin{array}{lcl}
n_0 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (), 0, M_{n_0,n_4}\} \\
n_1 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_0), 1, M_{n_0,n_4}\} \\
n_2 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_0, n_1), 2, M_{n_0,n_4}\} \\
n_3 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_1, n_2), 3, M_{n_0,n_4}\} \\
n_4 \rightarrow n_3 & : & \{\text{rrep}, n_0, n_4, n_0\#, n_4\#, 4, M_{n_4,n_0}, \\
 & & \quad (n_3), 0, (M^2_{n_4,n_3}, M^2_{n_4,n_2})\} \\
n_3 \rightarrow n_2 & : & \{\text{rrep}, n_0, n_4, n_0\#, n_4\#, 4, M_{n_4,n_0}, \\
 & & \quad (n_2, n_3, n_4), 1, (M^2_{n_4,n_2}, M^2_{n_3,n_1})\} \\
n_2 \rightarrow n_1 & : & \{\text{rrep}, n_0, n_4, n_0\#, n_4\#, 4, M_{n_4,n_0}, \\
 & & \quad (n_1, n_2, n_3), 2, (M^2_{n_3,n_1}, M^2_{n_2,n_0})\} \\
n_1 \rightarrow n_0 & : & \{\text{rrep}, n_0, n_4, n_0\#, n_4\#, 4, M_{n_4,n_0}, \\
 & & \quad (n_0, n_1, n_2), 3, (M^2_{n_2,n_0}, M^2_{n_1,n_0})\}
\end{array}$$

Figure 2: Operation example and messages of a route discovery with 2-*hop* crosscheck. $M_{n_0,n_4} = MAC_{n_0,n_4}(n_0\#)$**,** M_{n_4,n_0} **=** $MAC_{n_4,n_0}(rrep, n_0\#, n_4\#, 4)$**, and** $M^2_{n_i,n_j}$ **is computed as given in (5).**

In 2-*hop* crosscheck, if an intermediate node, say n_i, has a fresh route to destination n_l, when it receives a RREQ for destination n_l, n_i can send an authentication request message to its nexthop $FH^1_{n_i} = n_{i+1}$ and let n_{i+1} sends a RREP if it has valid route to the destination. When n_i receives the RREP from n_{i+1}, it can forward the RREP to n_{i-1} with authentication information from both n_{i+1} and itself.

Since RREQs and RREPs in DSR contain path information, 2-*hop* crosscheck can be applied to DSR more easily.

2.2.2 *complete* crosscheck

complete crosscheck can be applied to DSR directly since RREPs contain the whole path traversed by RREQs. Because the whole path list is contained in each RREP, the following fields are not necessary in the RREP sent or forwarded by node n_i: $BHL^l_{n_i}$, n_i, $FHL^l_{n_i}$, and fh_{n_i}. Therefore, the calculation of $M^l_{n_i,n_j}$ is changed as follows:

$$M^l_{n_i,n_j} = MAC_{n_i,n_j}(rrep, n_0, n_l, n_0\#, n_l\#, (n_1, \ldots, n_{l-1})) \qquad (6)$$

A *complete* crosscheck example is illustrated in Figure 3 and the messages in a route discovery are given in Figure 4.

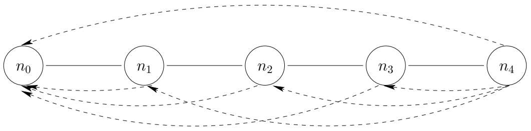

Figure 3: *complete* **crosscheck example (where l = 4): solid link indicates wireless link and dashed arc indicates node pairs that perform a crosscheck**

$$\begin{array}{lcl}
n_0 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (), M_{n_0,n_4}\} \\
n_1 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_1), M_{n_0,n_4}\} \\
n_2 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_1, n_2), M_{n_0,n_4}\} \\
n_3 \rightarrow * & : & \{\text{rreq}, n_0, n_4, n_0\#, (n_1, n_2, n_3), M_{n_0,n_4}\} \\
n_4 \rightarrow n_3 & : & \{\text{rrep}, n_0, n_4, n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, \\
 & & \quad (M^4_{n_4,n_3}, M^4_{n_4,n_2}, M^4_{n_4,n_1})\} \\
n_3 \rightarrow n_2 & : & \{\text{rrep}, n_0, n_4, n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, \\
 & & \quad (M^4_{n_4,n_2}, M^4_{n_4,n_1}, M^4_{n_3,n_0})\} \\
n_2 \rightarrow n_1 & : & \{\text{rrep}, n_0, n_4, n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, \\
 & & \quad (M^4_{n_4,n_1}, M^4_{n_3,n_0}, M^4_{n_2,n_0})\} \\
n_1 \rightarrow n_0 & : & \{\text{rrep}, n_0, n_4, n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, \\
 & & \quad (M^4_{n_3,n_0}, M^4_{n_2,n_0}, M^4_{n_1,n_0})\}
\end{array}$$

Figure 4: Operation example and messages of a route discovery with *complete* **crosscheck (l=4).** $M_{n_0,n_4} = MAC_{n_0,n_4}(n_0\#)$**,** M_{n_4,n_0} **=** $MAC_{n_4,n_0}(rrep, n_0\#, n_4\#, (n_1, n_2, n_3))$**, and** $M^4_{n_i,n_j}$ **is same as** M_{n_4,n_0} **and its calculation is given in (6).**

In order to prevent malicious nodes from modifying MAC values in the RREP, a cumulative MAC mechanism can be used. We use the same example in Figure 4 for illustration. All the MACs in a RREP are either authenticated by the destination or to be verified by the source. The MACs authenticated by the destination are computed as follows: $M^4_{n_4,n_1} = MAC_{n_4,n_1}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0})$, $M^4_{n_4,n_2} = MAC_{n_4,n_2}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, M^4_{n_4,n_1})$, $M^4_{n_4,n_3} = MAC_{n_4,n_3}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, M^4_{n_4,n_1}, M^4_{n_4,n_2})$.

Those MACs generated by intermediate nodes and verified by the source are computed as follows:
$M^4_{n_3,n_0} = MAC_{n_3,n_0}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0})$,
$M^4_{n_2,n_0} = MAC_{n_2,n_0}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, M^4_{n_3,n_0})$,
$M^4_{n_1,n_0} = MAC_{n_1,n_0}(n_0\#, (n_1, n_2, n_3), M_{n_4,n_0}, M^4_{n_3,n_0}, M^4_{n_2,n_0})$.

2.3 Discussion

2-*hop* crosscheck can be easily applied to AODV and DSR, and *complete* crosscheck can be applied to DSR easily since RREQ and RREP packets in DSR carry all the available path information. A 2-*hop* crosscheck can address route disruption attacks by the Active-1-1 adversary completely.

Number of Nodes	50
Node Mobility	Modified Random Waypoint
Node Minimum Speed	1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0
Node Maximum Speed	3.6, 7.1, 10.6, 14.1, 17.6, 21.1, 24.6, 28.2
Node Average Speed ($\overline{V}$)	2.0, 4.0, 6.0, 8.0, 10.0, 12.0, 14.0, 16.0
Pause Time	0 seconds
Field Size	1300 m × 800 m ($\rho = 10$) 900 m × 560 m ($\rho = 20$)
Radio Range	250 m
MAC	802.11
Routing protocol	AODV
Number of Traffic Pairs	20
Traffic Load	200, 400 Kbps (CBR/UDP)
Data Packet Payload	500 bytes
Link BW	2 Mbps
Hash length	128 bits
# of Attackers	0, 2, 4, or 8

Figure 5: Simulation Parameters. Node density, ρ, is the average number of nodes in a radio transmission area.

Since p-*hop* crosscheck enables intermediate nodes identify route falsification, its detection is more precise than the end-to-end authentication and verification mechanisms used by current secure routing protocols such as ARAN, SAODV, SRP, and Ariadne. For example, the 2-*hop* crosscheck can identify the location of a route falsifying node to a suspect group of 3 nodes (including the node that detects the misbehavior); complete crosscheck can identify the malicious node within a suspect group of 2 (including the detecting node). Therefore, p-*hop* crosscheck can be used to augment the existing secure routing protocols and intrusion detection techniques. For example, a 2-*hop* crosscheck or *complete* crosscheck can be used to complement Ariadne to prevent the Active-1-2 attack and Active-2-2 attack mentioned in [2].

The crosscheck mechanism can be used with public key cryptography if all pairs of nodes do not have shared keys. For example, the authentication node (say n_i) can authenticate the RREP using its private key and the verification node (say n_j) can use the public key of n_i to verify the message at the first time. Many techniques proposed in literature [4, 13, 7, 22] can be used to generate necessary shared keys between pairs of nodes. Then the authentication node and verification node can use the shared key instead.

3. PERFORMANCE EVALUATION

To see the security benefits of p-hop crosscheck, we conducted a detailed simulation analysis of the same. We implemented 2-*hop* crosscheck on AODV and used the Glomosim simulator, v2.03 [3] to evaluate the performance of 2-*hop* crosscheck with and without attacks. The simulation parameters used are listed in Figure 5. The modifications to random waypoint model for node mobility as given in [5, 19] are used to avoid clustering of nodes in the middle and gradual decay of average node speed. Node average speed ($\overline{V}$) is calculated according to [19]. We use golden rectangles (GRs) with length approximately 1.6 times the width ($1300 \times 800m^2$ and $900 \times 560m^2$). With 50 nodes, the node densities (ρ, the average number of nodes in a radio transmission area) are about 10 for the larger field and 20 for the smaller field.

We compared the performance of AODV with 2-*hop* crosscheck using the following metrics: (1) *Packet Delivery Ratio (PDR)* – the fraction of data packets sent that are received at the corresponding destination nodes; (2) *Packet Overhead* – the average number of transmissions of control packets per second, each hop-wise transmission of a control packet is counted as one transmission; (3) *Average end-to-end delay of data packets* – the average time elapsed from when a data packet is first sent to when it is first received at its destination. All experiments were run for 900 seconds. Each configuration was repeated 20 times and the results were averaged; the 95%-level confidence intervals are indicated for all data points.

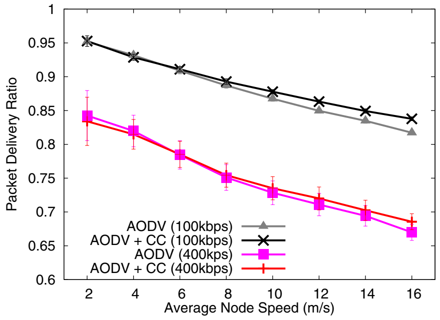

Figure 6: Packet delivery ratio vs. $\overline{V}$ in a normal network (ρ = 10).

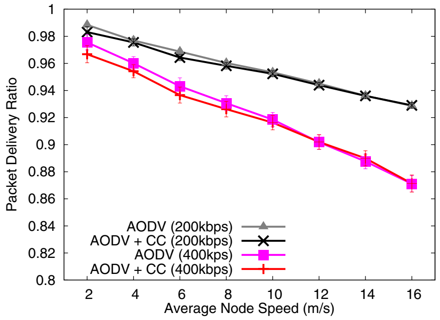

Figure 7: Packet delivery ratio vs. $\overline{V}$ in a normal network (ρ = 20).

In the first set of experiments, we evaluated the impact of the 2-*hop* crosscheck (denoted as CC in the simulation results) mechanism in a normal network without any attacks.

Figures 6 and 7 give the PDR of AODV with or without 2-*hop* crosscheck mechanism in both low and high node density networks. The original AODV and the AODV with crosscheck have nearly the same PDR. AODV with crosscheck has close average data packet delay as the original AODV with different average node speeds, as shown in Figures 8 and 9.

Figures 10 and 11 show the control packet overhead. AODV with and without crosscheck have nearly the same packet overhead with traffic load 200 kbps. When the traffic load increases to 400 kbps, AODV with crosscheck has slightly higher packet overhead. Since in AODV with crosscheck an intermediate node can not send RREP back to the source directly even if it has a fresh route, the route discovery takes longer with the crosscheck mechanism. Also, the use of crosscheck incurs higher byte Overhead (not shown).

In the second set of experiments, we evaluated the effectiveness of 2-*hop* crosscheck mechanism on AODV with attacks. We implemented a simple route disruption attack – black hole. Each malicious node sends a fabricated RREP with higher sequence number (indicating fresher route) or smaller hopcount (1 in our simulations) with 0.05 probability after receiving a RREQ. This lets a malicious node to put itself in more active routes than otherwise feasible, without generating excessive traffic that may be detected by an intrusion detection system. The malicious node drops all data packets going through it. Figures 12 and 13 give the PDR with and without crosscheck versus different number of attackers in

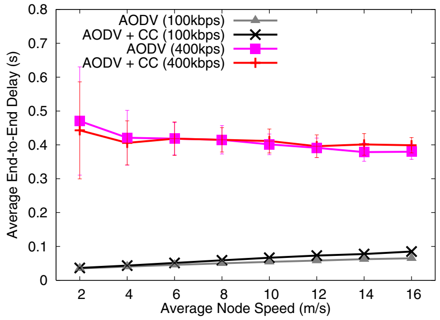

Figure 8: Average packet delay vs. $\overline{V}$ in a normal network (ρ = 10).

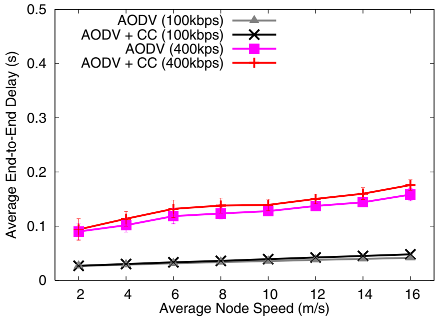

Figure 9: Average packet delay vs. $\overline{V}$ in a normal network (ρ = 20).

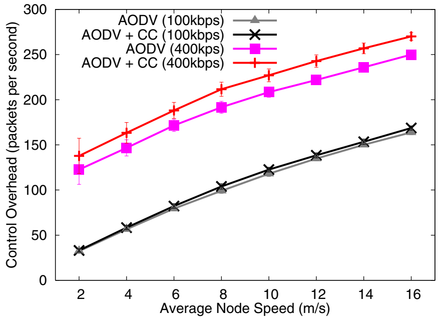

Figure 10: Packet overhead vs. $\overline{V}$ in a normal network (ρ = 10).

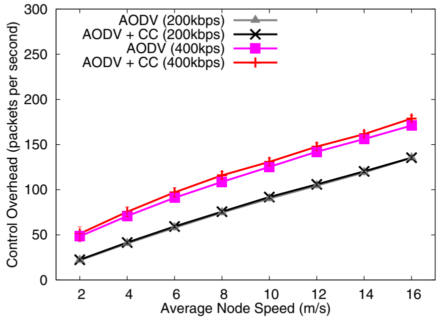

Figure 11: Packet overhead vs. $\overline{V}$ in a normal network (ρ = 20).

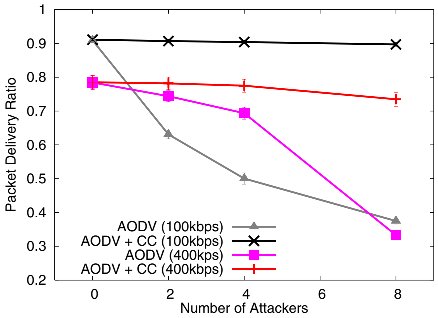

Figure 12: Packet delivery rate vs. number of attackers (ρ = 10, $\overline{V}$ = 6 m/s).

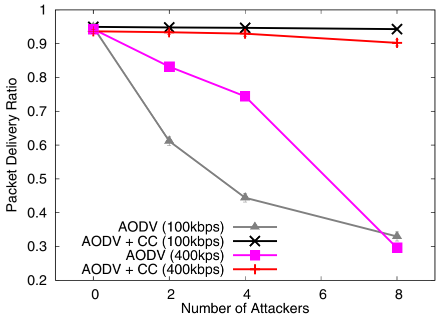

Figure 13: Packet delivery ratio vs. number of attackers (ρ = 20, $\overline{V}$ = 6 m/s).

low-density and high-density networks, respectively. The PDR of the original AODV drops dramatically with the number of attackers increasing, about 70% drop in high density network and 30% drop in low density network with 8 attackers. Whereas, the PDR of AODV with crosscheck changes only slightly when the number of attackers increases.

In summary, the proposed 2-*hop* crosscheck achieves the desirable security property with negligible performance impact.

4. RELATED WORK

Many secure on-demand routing protocols, such as ARAN [18], SAODV [20], SRP [14], Ariadne [10], SDSR [12], and endairA [2], are proposed for mobile ad hoc networks in the literature. ARAN and SAODV are based on AODV, while SRP, Ariadne, and endairA, and SDSR are based on DSR.

In ARAN, source signs the RREQ packets it initiates. Each node in the path verifies the signature of the previous node, replaces the signature of the previous (if it is not the source) with its signature of the packet, and retransmits it. So a RREQ contains, after one hop, two security codes. Destination verifies the signatures of its previous hop and the source. The security mechanisms used for RREPs are similar. However, ARAN involves too expensive computation since every message is signed in a point-to-point manner.

In SAODV, two security mechanisms are used: (i) digital signatures to authenticate the non-mutable fields of RREQs and RREPs and (ii) hash chains to secure the mutable information (hop count). Due to the shortcoming of one way hash chain mechanism, SAODV can not prevent a malicious node from forwarding a RREQ with the same hop count as in the RREQ it receives (replay attack). It also involves expensive computation to authenticate non-mutable fields of control packets using digital signatures.

The SRP [14] requires security verification only between source and destination of a route using MAC for RREQ and RREP packets. Since SRP does not require any authentication of the relay nodes in both route request and route reply stages, it makes the protocol more light-weight, but also more prone to attacks.

Ariadne [9] authenticates routing messages using one of three schemes: shared secret keys between each pair of nodes, shared secret keys between end-to-end nodes combined with broadcast authentication TESLA [16], or digital signatures. Though Ariadne can ensure that falsified route requests (or replies) are not accepted by the destination (or source) of the route being discovered, it cannot identify the nodes that caused the falsification.

An active adversary that control x adversarial nodes and uses y compromised identifiers is called an Active-y-x [9]. In [1], Acs *et al.* proposed two Active-1-1 attacks on SAODV. In [6], an Active-1-1 attack on SRP and an Active-1-1 attack on digital signature version of Ariadne were discussed. In [2], Acs *et al.* presented an Active-1-2 attack on Ariadne-MAC and an Active-2-2 attack on low-overhead version of Ariadne-MAC. In addition, SAODV, ARAN, SRP, and Ariadne suffers from a common problem that in the route reply stage each intermediate node does not verify RREPs (in SRP and Ariadne) or only verifies if messages comes from its next hop (in SAODV and ARAN). Single Active-1-1 attacker can fabricate a RREP easily, and other legitimate intermediate nodes can not detect this misbehavior and still relay the fabricated RREP until it arrives at the source. Even when the fabricated RREP is detected by the source, the source can not identify (or locate) the malicious node.

In endairA [2], the route request stage does not use any authentication. Intermediate nodes are involved in authentication and verification in the route reply stage. Every node (except destination) verifies the validity of digital signatures generated by all following nodes in the path list contained in the received RREP; then it (except source) signs the message and forwards the RREP to its preceding node in the path list. However, this involves expensive computations using digital signatures. Also, it can not detect falsification of RREQs.

Two results that are similar to p-hop crosscheck are the leap-frog protocol [8] and the interleaved hop-by-hop scheme [21]. The leap-frog protocol is based on a group shared key to secure broadcasts and multicasts. It is similar to our 2-hop crosscheck and can be used for static networks only. The interleaved hop-by-hop scheme filters injected false data in sensor networks, and assumes that falsification occurs in one direction: sensors to base station (equivalent to falsification of route replies from destination to source in ad hoc networks). On the other hand, p-hop cross check handles falsification of route requests, which are broadcasted, as well as route replies. Also, the *p-hop* crosscheck mechanism can be easily used to secure both table-driven routing protocol (AODV) and source routing protocols (DSR). Furthermore, we provide a performance analysis of the crosscheck mechanism.

5. CONCLUSIONS

In this paper, we proposed a simple *p-hop* crosscheck mechanism to identify malicious nodes falsifying route request and route reply control packets in on demand route discoveries. It addresses route falsification attacks by non-colluding attackers completely in a route discovery. The proposed general mechanism can be used to secure both on-demand table-driven routing protocols and source routing protocols such as AODV and DSR. It is simple and efficient using only pairwise symmetric keys. It can also be extended to situations that both pairwise symmetric keys and public and private keys exist. In addition, this mechanisms can also be used to complement existing secure routing protocols to address their attacks. We implemented 2-*hop* crosscheck on AODV in the Glomosim simulator. Using simulations, we showed that 2-*hop* crosscheck achieves desired security property with negligible performance impact.

In future, we intend to implement *p-hop* crosscheck on an ad hoc network testbed.

6. REFERENCES

[1] G. Ács, L. Buttyán, and I. Vajda. Provable security of on-demand distance vector routing in wireless ad hoc networks. In *Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, 2005.

[2] G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1546, 2006.

[3] R Bagrodia et al. Glomosim: A scalable network simulation environment, v2.03. Parallel Computing Lab, UC Los Angeles, CA, December 2000.

[4] R. Blom. An optimal class of symmetric key generation systems. In *Advances in Cryptology, EUROCRYPTą́r84, LNCS 209*, pages 335–338, 1984.

[5] Jean-Yves Le Boudec and Milan Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *Proceedings of IEEE INFOCOM*, pages 2743–2754, 2005.

[6] L. Buttyán and I. Vajda. Towards provable security for ad hoc routing protocols. In *Second ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, 2004.

[7] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proc. of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 27–31, 2003.

[8] M T. Goodrich. Leap-frog packet linking and diverse key distributions for improved integrity in network broadcasts. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 196–207, May 2005.

[9] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 12–23, 2002.

[10] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.

[11] David B Johnson, David A Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr). In *Internet Draft, draft-ietf-manet-dsr-09.txt*, April 2003.

[12] F. Kargl, A. Geiß, S. Schlott, and M. Weber. Secure dynamic source routing. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS)*, January 2005.

[13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proc. of 10th ACM Conference on Computer and Communications Security (CCS)*, 2003.

[14] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, Jan. 2002.

[15] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. *Ad Hoc On Demand Distance Vector (AODV) Routing*. IETF, July 2003. RFC 3561.

[16] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS)*, 2001.

[17] Jean-Francois Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Anonymity 2000, LNCS 2009*, pages 10–29, 2001.

[18] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A secure routing protocol for ad hoc networks. *Proceedings of IEEE ICNP*, 2002.

[19] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proceedings of ACM MobiCom*, 2003.

[20] M. G. Zapata. Secure ad hoc on-demand distance vector (SAODV) routing. In *IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-guerrero-manet-saodv-00.txt*, 2001.

[21] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *Proceedings of IEEE Symposium on Security and Privacy*, 2004.

[22] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of IEEE ICNP*, 2003.