A Secure Routing Protocol for Ad Hoc Networks

Bridget Dahill   Brian Neil Levine
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
{bridget,brian}@cs.umass.edu

Elizabeth Royer
Dept. of Computer Science
University of California
Santa Barbara, CA 93106
eroyer@cs.ucsb.edu

Clay Shields
Dept. of Computer Science
Georgetown Univeristy
Washington, DC 20007
clay@cs.georgetown.edu

August 28, 2001

Abstract

An ad hoc network is a collection of mobile nodes that establish and maintain connections purely over wireless connections; these networks do not have the infrastructure of a wired network. Because the nodes are mobile, links are continuously established and broken. An ad hoc routing protocol must therefore provide for dynamic routing, where new paths can be created as soon as old paths become obsolete. The problem with many proposed routing protocols for ad hoc networks is that these protocols have serious vulnerabilities to security attacks. In this paper, we demonstrate the specific exploits that are possible against existing ad hoc routing protocols. We propose a new protocol to provide secure routing for ad hoc networks that is based on authentication mechanisms.

1 Introduction

Ad hoc wireless networks assume no pre-deployed infrastructure is available for routing packets end-to-end in a network. Mobile nodes in an ad hoc network that are in radio range of one another communicate directly over wireless links; nodes which are too far apart to communicate directly must rely on intermediate nodes to route messages for them. Securing ad hoc routing presents challenges not present in traditional networks: neither centrally administered secure routers nor strict policy exist in an ad hoc wireless network. Furthermore, current ad hoc routing protocols inherently trust all participants. Users bring to an ad hoc environment their own mobile units, hope for cooperation, and fall under the threat of malicious adversaries.

In this paper, *we demonstrate exploits that are possible against ad hoc routing protocols and offer a secure solution with an authenticated routing protocol.* Many ad hoc routing protocols have been proposed previously [5, 7, 8, 9, 10, 11], however, none of the proposals have defined security requirements. We detail the exploits against two protocols in particular that are under consideration by the IETF for standardization: the Ad hoc On-demand Distance Vector routing protocol (AODV) [10] and the Dynamic Source Routing protocol (DSR) [5]. These exploits include *impersonation*, *modification*, and *fabrication* attacks. AODV and DSR are efficient in terms of network performance, but as is shown here, they allow attackers to easily advertise falsified route information, to redirect routes, and to launch denial-of-service attacks. Such exploits are difficult to deter in an ad hoc environment given the implicit trust among participants.

Our proposed protocol, Authenticated Routing for Ad hoc Networks (ARAN), detects and protects against malicious actions by third parties and peers in the ad hoc environment. ARAN introduces *authentication*, *message integrity*, and *non-repudiation* to an ad hoc environment as a part of a minimal security policy.

Applications that make use of ad hoc routing have heterogenous security requirements. We define three distinct environments that differ in their assumed pre-deployment and security requirements. Distinguishing these environments is important because satisfying a larger set of security requirements than an application requires is unwarranted and wasteful of resources.

Our work is a solution to the first of our three defined ad hoc network environments, the *managed-open* environment, which assumes a small opportunity for pre-deployed security infrastructure. We also define the following environments: *open* ad hoc wireless networks that have no opportunity for pre-deployment; and *managed-hostile* ad hoc wireless networks that have some pre-deployment and where annihilation of nodes or their users is a threat. We do not provide solutions to the open environment nor to the extreme security needs of the managed-hostile environment in this initial work, but our approach is promising for addressing those needs.

This paper is organized as follows. Section 2 overviews recent work on ad hoc routing protocols and reviews the basics of route acquisition in AODV and DSR. Section 3 presents the security exploits possible in ad hoc routing protocols. Section 4 presents our three defined ad hoc environments and the security requirements of any ad hoc network. Section 5 presents our secure ad hoc routing protocol, ARAN, and Section 6 offers concluding remarks.

2 Background

An ad hoc network forms when a collection of mobile nodes join together and create a network by agreeing to route messages for each other. There is no infrastructure in an ad hoc network, e.g., no centralized routers or administrative policy. Ad hoc routing protocols use the mobile nodes to route packets end-to-end. Since such ad hoc networks are also mobile, the network topology is dynamic; links are continuously established and broken due to movement of mobile nodes. In sum, conventional routing protocols do not work.

Several ad hoc routing protocols have been proposed [5, 7, 8, 9, 10]. All proposed protocols have security vulnerabilities and exposures that easily allow for routing attacks. Our goal is to define generic security requirements, but in this paper we focus on two protocols that are under consideration by the IETF for standardization: AODV and DSR. Below is a brief overview of route acquisition and maintenance basics in AODV and DSR. Detailed and authoritative descriptions of the AODV and DSR algorithms can be found elsewhere [10, 5].

The fundamental differences between ad hoc networks and standard IP networks necessitate the development of new security services. In particular, the measures proposed for IPSec [3] help only in end-to-end authentication and security between two network entities that already have routing between them; IPSec does not secure the routing protocol. While mechanisms similar to those used in IPSec can be adapted to secure the routing, IPSec alone does not suffice.

This point has been realized in other works that examine security problems in ad hoc networks. Zhou and Hass have proposed a solution that uses threshold cryptography as a mechanism for proving security to the network [16]. Hubaux, Buttyan and Capkun have proposed a method that is designed to ensure equal participation amoung members of the ad hoc group, and that gives each node the authority to issue certificates [4]. An effort to secure an existing ad hoc routing protocol has also recently been made available [15].

2.1 AODV

In AODV, each mobile node discovers or maintains routing information to another node if it is actively communicating with that node, or if it is an intermediary between two end points. If a node does not lie on an active path between two nodes, it does not maintain routing information for that path. AODV dynamically maintains loop-free routes, even when links change on active routes.

Route discovery in AODV is achieved with a source-initiated broadcast message called a *route request* (RREQ). When the RREQ reaches either the destination or an intermediate node that has a valid route to the destination, a *route reply* message (RREP) is unicast back to the source. As the RREP propagates back to the source, intermediate nodes receiving the RREP update their routing tables with a route to the destination.

AODV maintains fresh routes by implementing a counter for each node called a *sequence number*. A node's sequence number is incremented each time the local connectivity for that node changes. A RREQ contains fields for both the source and destination nodes' sequence numbers, called the *source_sequence_num* and *destination_sequence_num*, respectively. The RREP also contains a field for the *destination_sequence_num*. When an intermediate node receives a RREQ, it determines the freshness of its route table entry for the destination (if such an entry exists) by comparing the *destination_sequence_number* of the RREQ with that of its route table entry for the destination. The node then either responds to the RREQ with its own route to the destination (if that route is recent enough), or rebroadcasts the RREQ to its neighbors.

RREQ packets also contain a *broadcast_id* field. This value is a counter maintained at each node that is incremented every time the node initiates a new RREQ. The *broadcast_id*, together with the source node's IP address, serves as a unique identifier for a RREQ. Nodes can temporarily buffer this information for each received RREQ so that they can determine whether they have already processed a given RREQ.

2.2 DSR

DSR lets the sender determine the path packets travel toward a destination. This path is listed in the data packet header and is called a *source route*. Each node in the network maintains a dynamic *route cache* in which it stores routes to other nodes in the network that it has learned either from initiating a route request to a destination, or from forwarding packets along active paths for other nodes. In addition to the route discovery procedure, a node may also learn routes by overhearing transmissions of packets along paths for which it is not an intermediate node. This is called promiscuous listening.

When a node sends a packet to another node in the network, the sending node first checks its route cache for a source route to the destination node. If the sender has an entry for the destination node in its route cache, the sender inserts this source route into the packet's header, listing the addresses of nodes through which the packet will be forwarded on its path to the destination. The sending node transmits the packet to the first node on the route list. Upon receiving the packet, each intermediate node along the path forwards the packet to the next hop on the indicated path until the packet reaches the destination node.

If the sending node does not have a source route to the destination, it initiates the DSR *route discovery* process. This begins with a *route request* packet broadcast from the initiator node. Each route request packet is uniquely identified by the concatenation of the *initiator address*, the *target address*, and the *request id*. Upon receiving a route request packet, if the node is either the target node or has a route to the target node stored in its route cache, it responds with a *route reply* packet to the initiator. In the route reply is a *route record* of the string of nodes which constitute the path from initiator to target. If the receiving node does not have a route to the destination, it checks that it has not already processed this route request by checking the <*initiator address, request id*> pair of the route request. If it has processed this route request, the node discards the packet. Otherwise, if it is seeing this route request packet for the first time, the receiving node appends its address to the *route record* in the route request packet and forward broadcasts the packet.

3 Exploits allowed by existing protocols

The current proposed routing protocols for ad hoc wireless networks allow for many different types of attacks. In this section, we present and classify *modification*, *impersonation*, and *fabrication* exploits that are possible against current protocols. In Section 5, we propose a protocol that offers detection of and protection from these attacks.

Although IEEE 802.11, the commonly used link-layer standard for mobile networking, is susceptible to many security vulnerabilities [1, 2, 12], our focus is on vulnerabilities and exposures that result from the specification of the ad hoc routing protocol. Additionally, because participants of an ad hoc network also serve as routers, trivial denial-of-service attacks based on interception and non-cooperation are possible in all ad hoc routing protocols. While these attacks are possible, they are not achieved through subversion of the routing protocol.

We describe the attacks below in terms of the AODV and DSR protocols. These protocols are used as representatives of ad hoc on-demand protocols. Table 1 provides a summary of each protocol's vulnerability to the following exploits. Analogous exploits exist in wired networks [14], but are more easily defended against by infrastructure present in a wired network.

3.1 Attacks Using Modification

Current routing protocols assume that nodes do not alter the protocol fields of messages passed among nodes. Malicious nodes can easily cause redirection of network traffic and DoS attacks by simply altering these fields or by injecting routing messages into the network with falsified values in these fields. For example, in the network illustrated in Figure 1, a malicious node M could keep traffic from reaching X by consistently advertising to B a shorter route to X than the route to X which C is advertising. Below we detail several of the attacks possible if particular fields of routing messages in specific routing protocols are altered or falsified. Such attacks compromise the integrity of routing computations.

The first two attacks we present allow *remote redirection*, which allows an attacker to drop traffic, causing a denial of service attack, or forward the traffic on to a destination after viewing or altering the data payload. The third attack is a simple denial of service attack.

Attack	AODV	DSR
Remote redirection		
modification of sequence numbers	Yes	n/a
modification of hop counts	Yes	n/a
modification of source routes	n/a	Yes
Spoofing	Yes	Yes
Fabrication		
fabrication of error messages	Yes	Yes
fabrication of source routes (cache poisoning)	No	Yes

Table 1: Summary of vulnerabilities of AODV and DSR.

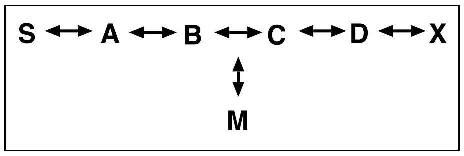


Figure 1: A simple ad hoc network.

3.1.1 Redirection with modified route sequence numbers

Protocols such as AODV and DSDV [9] instantiate and maintain routes by assigning monotonically increasing sequence numbers to routes toward specific destinations. We describe an attack on AODV that allows redirection of routes with the falsification of *destination sequence numbers*.

In AODV, any node may divert traffic through itself by advertising a route to a node with a *destination_sequence_num* greater than the authentic value. Figure 1 illustrates an example ad hoc network. Suppose a malicious node, M, receives the RREQ that originated from S for destination X after it is re-broadcast by B during route discovery. M redirects traffic towards itself by unicasting to B a RREP containing a significantly higher *destination_sequence_num* for X than the authentic value last advertised by X.

Eventually, the RREQ broadcast by B will reach a node with a valid route to X and a valid RREP will be propagated unicast back toward S. However, at that point B will have already received the false RREP from M. If the *destination_sequence_num* for X that M used in the false RREP is higher than the *destination_sequence_num* for X in the valid RREP, B will drop the valid RREP, thinking that the valid route is stale. All subsequent traffic destined for X that travels through B will be directed toward M. The situation will not be corrected until either a legitimate RREQ or a legitimate RREP with a *destination_sequence_num* for X higher than that of M's false RREP enters the network routing traffic.

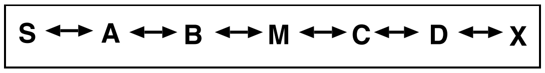


Figure 2: Another simple ad hoc network.

3.1.2 Redirection with modified hop counts

A redirection attack is also possible in certain protocols, such as AODV, by modification of the hop count field in route discovery messages. When routing decisions cannot be made by other metrics, AODV uses the hop count field to determine a shortest path. In AODV, malicious nodes can attract routes towards themselves by resetting the hop count field of the RREP to zero. Similarly, by setting the hop count field of the RREP to infinity, routes will tend to be created that do not include the malicious node. Such an attack is most threatening when combined with spoofing, as detailed in Section 3.2.

3.1.3 Denial of service with modified source routes

DSR is a routing protocol which explicitly states routes in data packets. These routes lack any integrity checks and a simple denial-of-service attack can be launched in DSR by altering the source routes in packet headers.

Assume a shortest path exists from node S to node X as in Figure 2. Also assume that nodes C and X cannot hear each other, that nodes B and C cannot hear each other, and that node M is a malicious node attempting a denial of service attack. Suppose node S wishes to communicate with node X and that S has an unexpired route to X in its route cache. S transmits a data packet toward node X, with the source route $S \rightarrow A \rightarrow B \rightarrow M \rightarrow C \rightarrow D \rightarrow X$ contained in the packet's header. When the malicious node M receives the packet, it can alter the source route in the packet's header, such as deleting node D from the source route. Consequently, when C receives the altered packet, it attempts to forward the packet to X. Since X cannot hear C, the transmission is unsuccessful.

DSR provides a route maintenance mechanism such that a node forwarding a packet is responsible for confirming that the packet has been received by the next hop along the path. If no confirmation of receipt is received after retransmitting the packet a specified maximum number of attempts, this node should return a route error message to the source node. In this case, C would send a route error message to S. Since M would be the first hop the route error takes on its path back to S, M can continue the denial-of-service attack by dropping this route error message.

DSR implements another route maintenance mechanism called *route salvaging* to recover from broken links along a path. When a break occurs, the node immediately upstream of the break can check its route cache, and if it has a different route to that destination, it can use that route instead. In our example C would check its route cache for an alternate route. If C only knows of the erroneous route to X, the DoS attack can be completed.

Modifications to source routes in DSR may also include the introduction of loops in the specified path. Although DSR prevents looping during the route discovery process, there are insufficient safeguards to prevent the insertion of loops into a source route after a route has been salvaged[1].

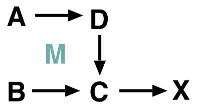
3.2 Attacks Using Impersonation

Current ad hoc routing protocols do not authenticate source IP addresses. By masquerading as another node, a malicious node can launch many attacks in a network. This is commonly known as *spoofing*. Spoofing occurs when a node misrepresents its identity in the network, such as by altering its MAC or IP address in outgoing packets. Spoofing is readily combined with modification attacks. The following example illustrates how this attack works in AODV. Similar attacks are possible in DSR as shown in Table 1.

[1]There is a potential for loops to form during route salvaging. An intermediate node salvaging the path replaces the source route in the packet with a new route from its route cache. DSR prevents infinite looping in this case by allowing a packet to only be salvaged a finite number of times.

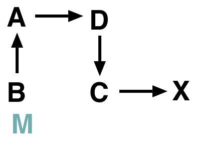
3.2.1 Forming loops by spoofing.

Assume a path exists between the four nodes illustrated below towards destination X as would follow after an AODV RREQ/RREP exchange:

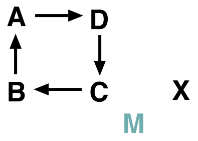

$$\begin{tikzcd}
A \arrow[r] & D \arrow[dd] & \\
M & & \\
B \arrow[r] & C \arrow[r] & X
\end{tikzcd}$$

In this example, A can hear B and D; B can hear A and C; D can hear A and C; and C can hear B, D, and X. M can hear all nodes except X.

A malicious attacker, M, can learn this topology exists by listening to the RREQ/RREP exchanges during route discovery. To start the attack, M changes its MAC address to match A's, moves closer to B and out of the range of A. It then sends a RREP to B that contains a hop count to X that is less than the one sent by C. B therefore changes its route to the destination to go through A, as illustrated below.


$$\begin{tikzcd}
A \arrow[r] & D \arrow[d] & \\
B \arrow[u] & C \arrow[r] & X \\
M & &
\end{tikzcd}$$

M then changes its MAC to match B's, moves closer to C and out of range of B, and then sends to D an RREP with a hop-count lower than what X had sent. C then routes through B.


$$\begin{tikzcd}
A \arrow[r] & D \arrow[d] & \\
B \arrow[u] & C \arrow[l] & X \\
& M &
\end{tikzcd}$$

At this point a loop is formed and X is partitioned from the routing. Our example shows how the attack is possible with a single malicious attacker, however, multiple attackers may collaborate for the same result.

3.3 Attacks Using Fabrication

We term the generation of false routing messages as fabrication attacks. Such attacks can be difficult to verify as valid constructs, especially in the case of fabricated error messages claiming that a neighbor cannot be contacted, as we discuss below.

3.3.1 Falsifying route error messages in AODV and DSR.

AODV and DSR implement path maintenance measures to recover broken paths when nodes move. If the source node moves, route discovery is re-initiated with a new route request message if the route is still needed. If the destination node or an intermediate node along an active path moves, the node upstream of the link break broadcasts a *route error* message to all active upstream neighbors. The node also invalidates the route for this destination in its routing table[2]. Upon receiving a route error message, if a node has an active route to this destination, and if the source of the route error message is the node's next hop along the path to the destination, the node deletes its routing table entry for the destination and forwards the route error message[3].

The vulnerability is that routing attacks can be launched by sending false route error messages. Suppose node S has a route to node X via nodes A, B, and C, as in Figure 1.

A malicious node M can launch a denial of service attack against X by continually sending route error messages to B spoofing node C, indicating a broken link between nodes C and X. B receives the spoofed route error message thinking that it came from C. B deletes its routing table entry for X and forwards the route error message on to A, who then also deletes its routing table entry. If M listens and broadcasts spoofed route error messages whenever a route is established from S to X, M can successfully prevent communications between S and X.

3.3.2 Route Cache Poisoning in DSR.

Corrupting routing state is also an attack against routing integrity, but is passive in nature. This occurs when information stored in routing tables at routers is either deleted, altered or injected with false information. Wired networks have been vulnerable to similar attacks [13, 6] but can often be defended against by security measures at routers. Since the routers in an ad hoc network are the nodes themselves, centralized defense is not possible.

Poisoning route caches is a common example of this attack. It follows the specifications of the routing protocol (wired or wireless) to *poison* caches at routers with false information. When a router forwards traffic to a destination listed in its routing tables, the router may unknowingly be forwarding the traffic to a false address. The following details such an attack in DSR.

In addition to learning routes from headers of packets which a node is processing along a path, routes in DSR may also be learned from promiscuously received packets. A node overhearing any packet may add the routing information contained in that packet's header to its own route cache, even if that node is not on the path from source to destination. For example, in Figure 1 a path exists from node S to node X via nodes A, B, and C. If a packet traveling along the source route from S to X is overheard by another node, that node may then add the route $<S,A,B,C,X>$ to its route cache.

The vulnerability is that an attacker could easily exploit this method of learning routes and poison route caches. Suppose a malicious node M wanted to poison routes to node X. If M were to broadcast spoofed packets with source routes to X via itself, neighboring nodes that overhear the packet transmission may add the route to their route cache.

Since this route discovery feature of caching overheard routing information is optional in DSR, this exploit can be easily patched by disabling this feature in the network. The downside of this is that without this feature DSR operates at a loss in efficiency.

4 Differentiating Security Requirements

Applications for ad hoc networks are numerous. They include military operations, emergency rescue missions, and simple provisioning of wireless network access, such as at a conference or in a classroom. In this section, we classify ad hoc networks into three distinct environments that differ in security needs and assumed pre-deployment. We define these classes because it is difficult to construct a single secure ad hoc routing protocol to suit the needs of many heterogenous applications. The lower security requirements of some environments do not justify use of costly protocols that satisfy stricter security policies. The environments we define in this section enable us to clearly state where we expect our secure protocol may be applied.

4.1 Three Environments

We define a set of three discrete ad hoc wireless environments: *open*, *managed-open*, and *managed-hostile.* The environments differ not only in the level of security needed, but also in that some environments may have opportunity for pre-deployed exchange of security parameters.

In an *open* environment, random nodes establish and maintain connectivity to other random nodes

[2]In DSR the source route is removed from the node's route cache.

[3]The exception in AODV occurs when the RERR message has the "No delete flag" set. In this case the node forwards the RERR without deleting its routing table entry.

without any trusted third parties in common. Since any node may be communicating with any other node for the first time, it cannot be expected that data or key exchange can occur in advance. This scenario might exist, for example, for a user walking through an urban environment or driving on a highway.

The *managed-open* environment is identical to the open environment except that nodes wishing to exchange communication may be able to exchange initialization parameters beforehand, perhaps within the security of an infrastructured network where session keys may be exchanged or through a trusted third party. Additionally, mobile nodes in the managed-open environment reside within some common context or geographic proximity. Such an ad hoc network might be formed by peers at a conference, or students on a campus.

A *managed-hostile* environment would be formed by military nodes in a battle environment, or perhaps by emergency response crews in a disaster area. In such an environment, nodes are deployed by a common source. Consequently, there may be opportunity for pre-deployed exchange of security parameters. The distinguishing security threat of the managed-hostile environment is that every node is vulnerable to physical *capture and take-over* of equipment, where hostile entities can then pose as friendly entities at a compromised node. Therefore, exposure of node location from the routing protocol messages is not desirable, else adversaries may gain an opportunity to annihilate users. Countermeasures for this threat must facilitate detection of compromised nodes. Means for this include other nodes sensing erratic behavior of the node, but care must be taken that nodes do not wrongfully accuse safe nodes of being compromised.

4.2 Security requirements of ad hoc networks

We define a number of fundamental security requirements for any ad hoc network. A good secure routing algorithm prevents each of the exploits presented in Section 3; it must ensure that no node can prevent successful route discovery and maintenance between any other nodes other than by non-participation. In sum, all secure ad hoc routing protocols must satisfy the following requirements to ensure that path discovery from source to destination functions correctly in the presence of malicious adversaries:

1. Route signalling cannot be spoofed;

2. Fabricated routing messages cannot be injected into the network;

3. Routing messages cannot be altered in transit, except for those changes that must be made according to the normal functionality of the routing protocol;

4. Routing loops cannot be formed through malicious action;

5. Routes cannot be redirected from the shortest path by malicious action.

The above requirements comprise the security needs of the open and managed-open environments. The security needs of these two environments are similar, however the following additional requirement distinguishes the managed-open environment:

6. Unauthorized nodes should be excluded from route computation and discovery;

Additionally, we assume that the managed-open environment has the opportunity for pre-deployment or exchange of public keys, session keys, or certificates.

We define the managed-hostile environment as having all the requirements listed above as well as the following:

7. The network topology must not be exposed neither to adversaries nor to authorized nodes by the routing messages.

Exposure of the network topology may be an advantage for adversaries trying to destroy or capture nodes.

In the next section, we present the ARAN protocol, which is satisfactory for the managed-open environment. It does not provide a solution to the open environment because it requires a trusted certificate server to operate correctly. It does not provide a solution to the managed-hostile environment because it exposes the routing topology. Providing solutions to these environments is the basis of our ongoing work.

ARAN implements public key cryptography and certificates to ensure secure routing in a managed-open environment. Our algorithm is split into two stages. The first stage is lightweight and guarantees requirements 1–5; however, it has the disadvantage that it allows malicious nodes on a path to lengthen routes on reverse path setup. The more expensive second stage of the algorithm ensures requirements 1–6, which includes secure shortest paths.

5 Authenticated Routing for Ad hoc Networks

In this section, we detail the operation of ARAN. Our security design is intended to manage the special circumstances common to ad hoc networks: quickly changing topologies, low-power devices, and heterogenous policy requirements of applications. Accordingly, ARAN is composed of two distinct stages. The first stage is simple and requires little extra work from peers beyond traditional ad hoc protocols. Nodes that perform the optional second stage increase the security of their route, but incur additional cost for their ad hoc peers who may not comply (e.g., if they are low on battery resources).

ARAN makes use of cryptographic certificates for the purposes of authentication and non-repudiation. These mechanisms allow ARAN to prevent most of the attacks presented in Section 3 and detect erratic behavior.

5.1 The ARAN Protocol

ARAN consists of a preliminary certification process, a mandatory end-to-end authentication stage, and an optional second stage that provides secure shortest paths. The optional stage is considerably more expensive than providing end-to-end authentication.

5.1.1 Certification

ARAN requires the use of a trusted certificate server T. Before entering the ad hoc network, each node requests a certificate from T. Details of how certificates are revoked are explained below. For a node A,

$$T \rightarrow A : \text{cert}_A = [IP_A, K_{A+}, t, e]K_{T-} \tag{1}$$

The certificate contains the IP address of A, the public key of A, a timestamp t of when the certificate was created, and a time e at which the certificate expires. Figure 3 summarizes our notation. These variables are concatenated and signed by T. All nodes must maintain fresh certificates with the trusted server and must know T's public key.

5.1.2 Stage 1: End-to-End Authentication

The goal of stage 1 is for the source to verify that the intended destination was reached. In this stage, the source trusts the destination to chose the return path.

The protocol is simple compared to most non-secured ad hoc routing protocols — we note that the exploits listed in Section 3 are primarily due to the optimizations that have been introduced into ad hoc routing protocols for route computation and creation.

A source node, A, begins route instantiation to a destination X by broadcasting to its neighbors a *route discovery packet* (RDP):

$$A \rightarrow \text{broadcast} : [\text{RDP}, \text{IP}_X, \text{cert}_A, N_A, t]K_{A-} \tag{2}$$

The RDP includes a packet type identifier ("RDP"), the IP address of the destination (IP_X), A's certificate (cert_A), a nonce N_A, and the current time t, all signed with A's private key. Each time A performs route discovery, it monotonically increases the nonce. The nonce and timestamp are used in conjunction with each other to allow for ease of nonce recycling. The nonce is made large enough that it will not need to be recycled within the probable clock skew between receivers. Nodes then store the nonce they have last seen with its timestamp. If a nonce later re-appears in a valid packet that has a later timestamp, the nonce is assumed to have wrapped around, and is therefore accepted. Note that a hop count is not included with the message.

Each node records the neighbor from which it received the message. It then forwards the message to each of its neighbors, signing the contents of the message. This signature prevents spoofing attacks that may alter the route or form loops. Let A's neighbor be B.

$$B \rightarrow \text{broadcast} : \\ [[\text{RDP}, \text{IP}_X, \text{cert}_A, N_A, t]K_{A-}]K_{B-}, \text{cert}_B \tag{3}$$

Nodes do not forward messages for which they have already seen the (N_A, IP_A) tuple. The IP address of A is contained in the certificate, and the monotonically increasing nonce facilitates easy storage of recently-received nonces.

Upon receiving the broadcast, B's neighbor C validates the signature with the given certificate. C then rebroadcasts the RDP to its neighbors, first removing B's signature.

$$C \rightarrow \text{broadcast} : \\ [[\text{RDP}, \text{IP}_X, \text{cert}_A, N_A, t]K_{A-}]K_{C-}, \text{cert}_C \tag{4}$$

Eventually, the message is received by the destination, X, who replies to the first RDP that it receives for a source and a given nonce. There is no guarantee that the first RDP received traveled along the shortest path from the source. The RDP that travels along the shortest path will not be received first only if it encounters congestion. In this case, however, a non-congested, non-shortest path is likely to

K_{A+}	Public-key of node A.
K_{A-}	Private-key of node A.
$\{d\}K_{A+}$	Encryption of data d with key K_{A+}.
$[d]K_{A-}$	Data d digitally signed by node A.
cert_A	Certficate belonging to node A.
t	timestamp.
e	Certificate expiration time.
N_a	Nonce issued by node A.
IP_A	IP address of node A.
RDP	Route Discovery Packet identifier.
REP	REPly packet identifier.
SPC	Shortest Path Confirmation packet identifier.
RSP	Recorded Shortest Path packet identifier.
ERR	ERRor packet identifier.

Figure 3: Table of variables and notation.

be preferred to a congested shortest path because of the reduction in delay. Because RDPs do not contain a hop count or specific recorded source route, and because messages are signed at each hop, malicious nodes have no opportunity to redirect traffic with the exploits we described in Section 3.

The destination unicasts a Reply (REP) packet back along the reverse path to the source. Let the first node that receives the RDP sent by X be node D.

$$X \rightarrow D : [\text{REP}, \text{IP}_a, \text{cert}_x, N_A, t]K_{X-} \quad (5)$$

The REP includes a packet type identifier ("REP"), the IP address of A (IP_a), the certificate belonging to X (cert_x), the nonce and associated timestamp sent by A. Nodes that receive the REP forward the packet back to the predecessor from which they received the original RDP. All REPs are signed by the sender. Let D's next hop to the source be node C.

$$D \rightarrow C : [[\text{REP}, \text{IP}_a, \text{cert}_x, N_A, t]K_{X-}]K_{D-}, \text{cert}_D \quad (6)$$

C validates D's signature, removes the signature, and then signs the contents of the message before unicasting the RDP to B.

$$C \rightarrow B : [[\text{REP}, \text{IP}_a, \text{cert}_x, N_A, t]K_{X-}]K_{C-}, \text{cert}_C \quad (7)$$

A node checks the signature of the previous hop as the REP is returned to the source. This avoids attacks where malicious nodes instantiate routes by impersonation and re-play of X's message.

When the source receives the REP, it verifies that the correct nonce was returned by the destination as well as the destination's signature.

Only the destination can answer an RDP packet. Other nodes that already have paths to the destination cannot reply for the destination. While other protocols allow this networking optimization, we note that removing it also removes several possible exploits and cuts down on the reply traffic received by the source. If reliability of the REP returning to the source is a concern, however, the IEEE 802.11 standard can be relied on to provide hop-by-hop reliability. If necessary, a timeout at the source can trigger a new RDP broadcast. Furthermore, because only the destination can send REPs, loop freedom is guaranteed easily. ARAN is not susceptible to spoofed REPs that can redirect routing or form loops.

ARAN requires that nodes keep one routing table entry per source-destination pair that is currently active. This is certainly more costly than per-destination entries in non-secure ad hoc routing protocols. We will address this overhead in future revisions of the protocol.

5.1.3 Stage 2: Secure Shortest Paths

The first stage of ARAN does not guarantee in a secure fashion that the instantiated route is the shortest path. The optional second stage of the protocol ensures shortest paths but is very costly. Sources must first perform the first stage of ARAN in order to learn the certificate of the destination. However, route instantiation has already taken place at the end of stage one and data transfer may begin while stage 2 occurs.

The source begins by broadcasting a *Shortest Path Confirmation* (SPC) message to its neighbors (the same variables are used as in stage 1.)

$$A \rightarrow \text{broadcast} :$$
$$SPC, \text{IP}_X, \text{cert}_X, \{[\text{IP}_X, \text{cert}_A, N_A, t]K_{A-}\}K_{X+}$$

$$\tag{8}$$

The SPC message begins with the SPC packet identifier ("SPC"), X's IP address and certificate. The source concatenates a signed message containing the IP address of X, its certificate, a nonce and timestamp. This signed message is encrypted with X's public key so that other nodes cannot modify the contents.

A neighbor that receives the message, B, rebroadcasts the message after including its own cryptographic credentials. B signs the encrypted portion of the received SPC, includes it's own certificate, and re-encrypts with the public key of X. This public key can be obtained in the certificate forwarded by A. This operation is equivalent to B recording its address in the packet as in DSR and also serves as a hop count as in AODV. We assume that the encryption method used by B precludes any adversairies from removing B's credentials by simply deleting to the last portion of the encrypted message.

$$\begin{aligned} B \rightarrow \text{broadcast} : &\ \text{IP}_X, \text{cert}_X, \\ &\ SPC, \text{IP}_X, \text{cert}_X, \\ &\ \{[\{[\text{IP}_X, \text{cert}_A, N_A, t]K_{A-}\}K_{X+}]K_{B-}, \text{cert}_B\}K_{X+} \end{aligned} \tag{9}$$

Just as in stage 1, nodes that receive the SPC packet create entries in their routing table so as not to forward duplicate packets. The entry also serves to route the reply packet from the destination along the reverse path.

The onion-like signing of messages prevents nodes in the middle from changing the path in several ways. First, to increase the path length of the SPC, malicious nodes require an additional valid certificate. Second, malicious nodes cannot decrease the recorded path length or alter it because doing so would break the integrity of the encrypted data.

Once the destination X receives the SPC, it checks that all the signatures are valid. X replies to the first SPC it receives and also any SPC with a shorter recorded path. X sends a Recorded Shortest Path (RSP) message to the source through its predecessor D.

$$X \rightarrow D : [\text{RSP}, \text{IP}_A, \text{cert}_X, N_A, \text{route}]K_{X-} \tag{10}$$

The SPC includes a specific route on which the SPC must travel. Each node forwards the message without modification or addition along the route stated in the packet. Nodes may drop RSP for which they are not listed. The source eventually receives the packet and verifies the nonce corresponds to the SPC it originally generated.

5.2 Route Maintenance

ARAN is an on-demand protocol. Nodes keep track of whether routes are active. When no traffic has occurred on an existing route for that route's lifetime, the route is simply de-activated in the route table. Data received on an unactive route causes nodes to generate an Error (ERR) message that travels the reverse path towards the source. Nodes also use ERR messages to report links in active routes that are broken due to node movement. All ERR message must be signed. For a route between source A and destination X, a node B generates the ERR message for its neighbor C as follows:

$$B \rightarrow C : [ERR, \text{IP}_A, \text{IP}_X, \text{cert}_c, N_b, t]K_B \tag{11}$$

This message is forwarded along the path towards the source without modification. A nonce and timestamp ensures the ERR message is fresh.

It is extremely difficult to detect when ERR messages are fabricated for links that are truly active and not broken. However, because messages are signed, malicious nodes cannot generate ERR messages for other nodes. The non-repudiation provided by the signed ERR message allows a node to be verified as the source of each ERR message that it sends. A node which transmits a large number of ERR messages, whether the ERR messages are valid or fabricated, should be avoided. Though beyond the scope of this initial paper, such information may be used to avoid error-prone nodes during route creation.

5.3 Key Revocation

As with any secure system based on cryptographic certificates, the problem exists of making sure that certificates are able to be revoked when the holder is no longer authorized to access the system. In some environments, with strict security criteria, this revocation mechanism can be very reliable and expensive. Due to the desired low-overhead in wireless networks, and to the lower standards of security sought in the open-managed environment, we provide a best-effort immediate revocation service that is backed up by the use of limited-time certificates.

In the event that a certificate needs to be revoked, the trusted certificate server, T, sends a broadcast message to the ad hoc group that announces the revocation. Calling the revoked certificate $cert_r$, the transmission appears as:

$$T \rightarrow \text{broadcast} : [revoke, \text{cert}_r]K_{T-} \tag{12}$$

Any node receiving this message re-broadcasts it to its neighbors. Revocation notices need to be stored until the revoked certificate would have expired normally. Any neighbor of the node with the revoked certificate needs to reform routing as necessary to avoid transmission through the now-untrusted node.

This method is not failsafe. In some cases, the untrusted node that is having its certificate revoked may be the sole connection between two parts of the ad hoc network. In this case, the untrusted node may not forward the notice of revocation for its certificate, resulting in a partition of the network, as nodes that have received the revocation notice will no longer forward messages through the untrusted node, while all other nodes depend on it to reach the rest of the network. This only lasts as long as the untrusted node's certificate would have otherwise been valid, or until the untrusted node is no longer the sole connection between the two partitions. At the time that the revoked certificate should have expired, the untrusted node is unable to renew the certificate, and routing across that node ceases. Additionally, to detect this situation and to hasten the propagation of revocation notices, when a node meets a new neighbor, it can exchange a summary of its revocation notices with that neighbor; if these summaries do not match, the actual signed notices can be forwarded and re-broadcasted to restart propagation of the notice.

6 Conclusion

In this paper we have completed many of the steps toward securing ad hoc networks. By classifying ad hoc networks according to the security threats of the environment and whether there is opportunity for predeployment, we can create a generic ad hoc routing protocol with optional features that are enabled for a particular environment. In this paper we have presented a protocol which defends the managed-open networks against their specific security threats. With authentication assured, secure routing can be successful in an ad hoc network and nodes can identify erratic nodes and exclude them from routing if necessary. We plan to continue this work by expanding the ARAN protocol to provide secure routing for the managed-hostile environment, as well as secure key exchange for the open networks.

References

[1] W. Arbaugh, N. Shankar, and Y.C. Wan. Your 802.11 wireless network has no clothes. Technical report, Dept. of Computer Science, University of Maryland, March 2001.

[2] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html.

[3] C. R. Davis. *IPSec: Securing VPNs.* McGraw-Hill, New York, NY, USA, 2000.

[4] Jean-Pierre HuBaux, Levente Buttyan, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing*, October 2001.

[5] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. *IEEE Internet Draft*, March 2001. draft-ietf-manet-dsr-05.txt (work in progress).

[6] K. Knox. Ip (routing information protocol) version 1 spoofer. http://rootshell.com/archive-j457nxiqi3gq59dv/199711/rip.c.html, 1996.

[7] S. Murthy and J. J. Garcia-Lunca-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, pages 183–197, October 1996. Special Issue on Routing in Mobile Communication Networks.

[8] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOMM '97*, April 1997.

[9] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *Computer Communications Review*, pages 234–244, October 1994.

[10] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[11] Elizabeth M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[12] A. Stubblefield, J. Ioannidis, and A. D. Rubin. Using the fluhrer, mantin, and shamir attack to break wep. Technical Report TD-4ZCPZZ, AT&T Labs, August 2001.

[13] Mahesh Tripunitara and Partha Dutta. A Middleware Approach to Asynchronous and Backward-Compatible Detection and Prevention of ARP Cache Poisoning. In *Proceedings 15th Annual Computer Security Applications Conferen ce (ACSAC'99)*, pages 303–309, December 1999.

[14] Feiyi Wang, Brian Vetter, and Shyhtsun Felix Wu. Secure routing protocols: Theory and practice. Technical report, North Carolina State University, May 1997.

[15] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad hoc routing for wireless networks. Technical Report UIUCDCS-R-2001-2241, UILU-ENG-2001-1748, University of Illinois at Urbana-Champaign, August 2001.

[16] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.