Secure Ad-Hoc Routing Protocols

Mehul K Revankar

Abstract—An Ad-Hoc network is a collection of mobile nodes dynamically forming a temporary network without the presence of existing network infrastructure. A lot of research has taken place in developing efficient routing protocols for the Ad-Hoc Network, but very little has been done to incorporate security feature into the protocol. In this paper I have done a detailed study of three protocols namely SEAD (Secure Efficient Distance Vector Routing Protocol for Ad-Hoc Network), ARAIDNE (A secure on demand Routing protocol for Ad-Hoc Networks) and ARAN (A Routing protocol for Ad-hoc Networks) and I have tried to examine the mechanisms that have been implemented to defend against well known attacks. I have also tried to evaluate the performance overhead within the respective protocols due to inclusion of the security feature.

I. INTRODUCTION

In a mobile wireless ad hoc network, computers (nodes) in the network cooperate to forward packets for each other, due to the limited wireless transmission range of each individual node. The network route from some sender node to a destination node may require a number of intermediate nodes to forward packets to create a “multihop” path from this sender to this destination. The role of the routing protocol in an ad hoc network is to allow nodes to learn such multihop paths. Since the nodes in the network may move at any time, or may even move continuously, and since sources of wireless interference and wireless transmission propagation conditions may change frequently, the routing protocol must also be able to react to these changes and to learn new routes to maintain connectivity.

Ad hoc networks require no centralized administration or fixed network infrastructure such as base stations or access points, and can be quickly and inexpensively set up as needed. They can thus be used in scenarios where no infrastructure exists, or where the existing infrastructure does not meet application requirements for reasons such as security, cost, or quality. Examples of applications for ad hoc networks range from military operations and emergency disaster relief, to community networking and interaction between attendees at a meeting or students during a lecture. In these and other applications of ad hoc networking, security in the routing protocol is necessary in order to guard against attacks such as malicious routing misdirection, but relatively little previous work has been done in securing ad hoc network routing protocols.

Secure ad hoc network routing protocols are not easy to design, due to the generally highly dynamic nature of an ad hoc network and due to the need to operate efficiently with resources, including network bandwidth and the CPU processing capacity, memory, and battery power (energy) of each individual node in the network. Existing insecure ad hoc network routing protocols are often highly optimized to spread new routing information quickly as conditions change, requiring more rapid and often more frequent routing protocol interaction between nodes than is typical in a traditional (e.g., wired and stationary) network. Expensive and cumbersome security mechanisms can delay or prevent such exchanges of routing information, leading to reduced routing effectiveness, and may consume excessive network or node resources, leading to many new opportunities for possible Denial-of-Service (DoS) attacks through the routing protocol.

In this paper I have tried to examine the security mechanisms that have been implemented in SEAD [1], ARIADNE [2] and ARAN [3] to guard against well-known attacks in Ad-Hoc Routing Protocols. I have also tried to evaluate their security and performance with the existing protocols.

2. Routing Protocols

Routing involves transfer of information over the network from a source router and destination router. In Ad-hoc networks mobile nodes acts as a router as well as a host. Mobile nodes use routing algorithms to find the best route to a destination. When we say "best route," we consider parameters like the number of hops (the trip a packet takes from one router or intermediate point to
Secure Ad-Hoc Routing Protocols

Routing protocols historically have been divided into two kinds of protocols: Distance Vector Protocol and Link State Routing Protocol.

**Distance Vector Protocol** [8]:
This type of routing protocol requires each router simply to inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement.

**Link State Routing Protocol** [9]:
This type of routing protocol requires each router to maintain at least a partial map of the network. When a network link changes state (up to down, or vice versa), a notification, called a *link state advertisement* (LSA) is *flooded* throughout the network. All the routers note the change, and recompute their routes accordingly. This method is more reliable, easier to debug and less bandwidth-intensive than Distance-Vector. It is also more complex and more compute- and memory-intensive.

### 2.1 Ad-Hoc Routing Protocols

The drawback of the above mentioned protocols is that they require periodic updates of the network topology from all the nodes in the network. This kind of behavior is extremely expensive on a resource constrained mobile ad-hoc network node. This has given rise to a new breed of protocols called *on-demand protocols*. In an on-demand protocol, nodes exchange routing information only when needed, with a node attempting to discover a route to some destination only when it has a packet to send to that destination. Given below is a list of on-demand routing protocols.

**DSR** [6]: *Dynamic Source Routing* – is based on source routing and the sender knows the complete hop-by-hop route to the destination. When the mobile station in network has a packet to send to some destination and does not know the route, it initiates *route discovery* process to determine the route. It locally broadcasts a route request packet and the destination or any node with the route sends a reply with the route. If any link on a source route is broken, the source node is notified using a route error packet. DSR can be used with some optimization like broadcast authentication of control messages.

**AODV** [10]: *Ad Hoc On-Demand Distance Vector Routing* – AODV has on-demand characteristics and discovers routes when needed via a route discovery process. However, it maintains only one entry per destination whereas DSR keeps multiple route cache entries for each destination. AODV uses sequence numbers maintained at each destination to determine freshness of routing information and to prevent loops. It also keeps a timer with each entry in routing table and deletes the entry if not recently used.

Ad-hoc routing Protocols also include distance vector protocols such as DSDV. DSDV is an enhanced version of distance vector protocol which uses sequence numbers and weighted settling time to trigger updates. This feature essentially reduces the number of routing updates.

**DSDV** [11]: *Destination-Sequenced Routing Protocols* – is a distance vector protocol based on the idea of the classical Bellman-Ford routing algorithm. Every mobile station (router) maintains a routing table that lists all the available destinations and the number of hops to reach them. Each station advertises its routing table to its neighbor periodically and also sends triggered updates when a significant change has occurred in its table. It also uses sequence numbers to distinguish between stale routes from new ones and thus avoid the formation of loops.

All the above mentioned protocols are extremely efficient in routing information from one mobile Ad-hoc node to another ad-hoc node. A number of optimizations have also been proposed for the above protocols which further enhance their efficiency. However all the above protocols have been designed for a trusted environment i.e. there has been no consideration for security of information that is being routed by the mobile node.

### 3. Security in Ad-Hoc Networks

It is extremely difficult to implement security in Ad-hoc networks because Ad-hoc mobile nodes have limited resources in the form of CPU processing, memory and battery power. Further the ad-hoc network itself has a limitation on the network bandwidth. Typically security services can be thought of to have 5 key attributes i.e. **Authenticity, Confidentiality, Integrity, Availability and Non-Repudiation**, but interestingly when we discuss Ad-hoc routing protocols only 3 among the 5 key attributes...
namely Authenticity, Integrity and Availability need to be satisfied. This point can justify by pointing out the fact that most of well known attacks discussed in section 3.1 are due to fabrication or alteration of data in transit or due to impersonation of a node by a malicious node.

3.1 Attacks in Ad-hoc Network

All the Protocols discussed above are vulnerable to many different types of attacks. In this section, I would like to list different types of attacks that are possible in Ad-Hoc networks.

3.1.1 Attacks Using Modification
An attacker node may modify certain contents of the routing packet, thus propagating incorrect information in the network.

3.1.2 Attacks Using Impersonation
A malicious node may try to impersonate a node and send data on its behalf. This attack is generally used in combination with modification attack.

3.1.3 Attacks Using Fabrication
An attacker may try to fabricate a false Route Error message, which may cause other nodes to remove a particular node from its routing table.

3.1.4 Packet Dropping

Black Hole
An attacker may create a routing black hole, in which all packets are dropped. By sending forged routing packets, the attacker could route all packets for some destination to itself and then discard them.

Gray Hole
As a special case of a black hole, an attacker could create a gray hole, in which it selectively drops some packets but not others, for example, forwarding routing packets but not data packets.

3.1.5 Gratuitous detour
An attacker may attempt to make a route through itself appear longer by adding virtual nodes to the route. The attack is named so as a shorter route exists and would otherwise have been used.

3.1.6 Partition of the network
An attacker may attempt to partition the network by injecting forged routing packets to prevent one set of nodes from reaching another.

4. Secure Ad-Hoc routing protocols

I would now like to list the three protocols that I have studied and discuss the mechanisms that they have implemented to guard against the above mentioned attacks. It would become obvious in the following sections that it is extremely difficult to guard against all the above attacks; however attempts have been made to at least reduce the impact of certain attacks.

4.1 ARIADNE
ARIADNE is a security extension of DSR protocol. It uses a symmetric cryptosystem with an asymmetric primitive. It has been designed to incorporate Authenticity and Integrity in DSR protocol. It uses TESLA [5] to provide integrity of the routing information and symmetric cryptosystem to provide authenticity.

4.1.2 Overview of TESLA
TESLA (Time Efficient Stream Loss-Tolerant Authentication) is a broadcast authentication protocol. TESLA is efficient and adds only a single message authentication code (MAC) to a message for broadcast authentication. For broadcast communication, multiple receivers need to know the MAC key for verification, which would allow any receiver to forge packets and impersonate the sender. Secure broadcast authentication thus requires an asymmetric primitive, such that the sender can generate valid authentication information, but the receivers can only verify the authentication information. TESLA achieves this asymmetric primitive using One-way hash chain [12] and delayed key disclosure with time synchronization. One way hash key chain is generated by repeatedly applying a hash function to a random value.

One-way hash chains:
Figure 1:
$S_0 = \text{Random Value.}$
$F() = \text{Hash Functions.}$
$S_0 = \text{Commitment of the chain.}$

### 4.1.2.1 Time Synchronization:

TESLA requires all the nodes in the network to have loose time synchronization between all the nodes. This criteria essentially means that each node should be aware about the latest time any particular node could be in with respect to its own time in the network.

To use TESLA, each sender chooses a random key $S_i$ and generates a one-way key chain $(S_0, S_1, \ldots, S_l)$ by repeatedly computing a one-way hash function $H$ by:

$$K_{S,i} = H(K_{S,i-1})$$

TESLA relies on a receiver’s ability to determine which keys a sender may have already published based on loose time synchronization. Each sender predetermines a schedule for publishing key in reverse order. If $\Delta$ is maximum time synchronization error between two nodes knows to all nodes, $\tau$ is the pessimistic upper bound on end-to-end network delay then sender could choose a key $K_{\tau}$ which will not publish until time $\tau + 2\Delta$ in future. In sending the packet, the sender adds a MAC using key $K_i$. When the packet reaches the receiver, it rejects the packet if the key $K_i$ is published by that time. If the key hasn’t been published then it waits for key to be published and then authenticates the message.

### 4.1.2 Ariadne basic design

#### 4.1.2.1 Assumptions

The basic design of ARIADNE assumes all the nodes in the network are aware of the time synchronization error $\Delta$. It assumes some means to setup shared secret keys and share commitment of one-way hash chains.

#### 4.2.1 Notations:

The following notation to describe security protocols and cryptographic operations:

- $A, B$ are principals, such as communicating nodes.
- $K_{ab}, K_{ba}$ denote the secret MAC keys shared between $A$ and $B$ (one key for each direction of communication).
- $\text{MAC } K_{ab}(M')$ denotes the computation of the message authentication code (MAC) of message $Z$ with the MAC key $K_{ab}$.

#### 4.2.2 Ariadne Operation

The operation of Ariadne can be summarized in three simple steps:

1. Authentication of Target
2. Authentication of Data in Route Requests by any of the following.
   - TESLA
   - Digital Signatures
   - MACs
3. A mechanism to verify that no node is missing.
   - Per Hop Hashing.

### 4.2.3 Route Discovery in Ariadne

Route Discovery with Tesla

A ROUTE REQUEST packet in Ariadne contains 8 fields:

ROUTE REQUEST, initiator, target, id, time interval, hash chain, node list, MAC list. The initiator and target are set to the address of the initiator and target nodes, respectively. The source sets the id to an identifier that it has not recently used in initiating a Route Discovery. The time interval is the TESLA time interval at the pessimistic expected arrival time of the REQUEST at the target.
When a node source $S$ wants to send a packet to destination $D$, to which it does not have a route, $S$ initiates route discovery and sends a ROUTE REQUEST packet to $D$ along with a MAC computed with the shared secret key. All the intermediate nodes append its own name to the node list and authenticate the data by computing a hash value of the message and generating a MAC with undisclosed TESLA secret key for the message. The hash and MAC are then appended to the message and the packet is forwarded. When the target receives the route discovery packet it checks whether any of the undisclosed keys have been disclosed and if no, it checks whether the hash value is valid if yes, it replies back to the source with a ROUTE REPLY packet along with a MAC of the entire message. On the way back all the intermediate nodes disclose their keys which are in turn validated by the source. Given below is an example to show the route discovery process in Ariadne.

\[ S : h_0 = \text{MACKSD}(\text{REQUEST}, S, D, id, ti) \]

\[ S \rightarrow*: \quad _{\text{REQUEST}, S, D, id, ti, h_0, (), ()} \]

\[ : h_1 = H[A, h_0] \]

\[ \text{MA} = \text{MACKAti}(\text{REQUEST}, S, D, id, ti, h_1, (A), ()) \]

\[ A \rightarrow*: \quad _{\text{REQUEST}, S, D, id, ti, h_1, (A), (\text{MA})} \]

\[ B : h_2 = H[B, h_1] \]

\[ \text{MB} = \text{MACKBti}(\text{REQUEST}, S, D, id, ti, h_2, (A,B), (\text{MA})) \]

\[ B \rightarrow*: \quad _{\text{REQUEST}, S, D, id, ti, h_2, (A,B), (\text{MA,MB})} \]

\[ C : h_3 = H[C, h_2] \]

\[ \text{MC} = \text{MACKCti}(\text{REQUEST}, S, D, id, ti, h_3, (A,B,C), (\text{MA,MB})) \]

\[ C \rightarrow*: \quad _{\text{REQUEST}, S, D, id, ti, h_3, (A,B,C), (\text{MA,MB,MC})} \]

\[ D : \quad \text{MD} = \text{MACKDS}(\text{REPLY}, D, S, ti, (A,B,C), (\text{MA,MB,MC})) \]

\[ D \rightarrow C : \quad _{\text{REPLY}, D, S, ti, (A,B,C), (\text{MA,MB,MC}),\text{MD}, ()} \]

\[ C \rightarrow B : \quad _{\text{REPLY}, D, S, ti, (A,B,C), (\text{MA,MB,MC}),\text{MD}, (K\text{Cti})} \]

\[ B \rightarrow A : \quad _{\text{REPLY}, D, S, ti, (A,B,C), (\text{MA,MB,MC}),\text{MD}, (K\text{Cti}, K\text{Bti})} \]

\[ A \rightarrow S : \quad _{\text{REPLY}, D, S, ti, (A,B,C), (\text{MA,MB,MC}),\text{MD}, (K\text{Cti}, K\text{Bti}, K\text{Ati})} \]

### 4.1.2.4 Route Maintenance:

A node generates and sends a ROUTE ERROR packet original sender of the packet if it is unable to deliver the packet to the next hop after a limited number of retransmission attempts. To prevent unauthorized nodes from sending ERRORS, it is required that an ERROR be authenticated by the sender. Each node on the return path to the source forwards the ERROR. If the authentication is delayed, for example when TESLA is used, each node that will be able to authenticate the ERROR buffers it until it can be authenticated.

A ROUTE ERROR packet in Ariadne contains six fields:

- ROUTE ERROR, sending address, receiving address, time interval, error MAC, recent TESLA key. The sending address is set to the address of the intermediate node encountering the error, and the receiving address is set to the intended next hop destination of the packet it was attempting to forward. The time interval in the ROUTE ERROR is set to the TESLA time interval at the pessimistic expected arrival time of the ERROR at the destination, and the error MAC field is set to the MAC of the preceding fields of the ROUTE ERROR, computed using the sender of the ROUTE ERROR’s TESLA key for the time interval specified in the ERROR. The recent TESLA key field in the ROUTE ERROR is set to the most recent TESLA key that can be disclosed for the sender of the ERROR. When sending a ROUTE ERROR, the destination of the packet is set to the source address of the original packet triggering the ERROR, and the ROUTE ERROR is forwarded toward this node in the same way as a normal data packet. Each node that is either the destination of the ERROR forwards the ERROR searches its Route Cache for all routes it has stored that use the d sending address, receiving address link indicated by the ERROR. If the node has no such routes in its Cache, it does not process the ROUTE ERROR further (other than forwarding the Packet, if it is not the destination of the ERROR). Otherwise, the node checks whether the time interval in the ERROR is valid that time interval must not be too far into the future and the key corresponding to it must not have been disclosed yet; if the time interval is not valid, the node similarly does not process the ROUTE ERROR further.

If all of the tests above for the ROUTE ERROR succeed, the node checks the authentication on the ERROR, based on the sending node’s TESLA key for the time interval indicated in the ERROR. To do so, the node saves the information from the ERROR in memory until it receives a disclosed TESLA key from the sender that allows this. During this time, the node continues to use the routes in its Route Cache without modification from this ERROR. The routes are not removed from the routing tables unless the ROUTE ERROR has been authenticated. Once the Route error is authenticated all the nodes remove the route from their routing tables.
4.2 ARAN

ARAN makes use of cryptographic certificates to offer routing security. It requires that each node that needs to communicate in the network to have a valid certificate from a trusted server. This approach essentially prevents any third party node to take part in any kind of communication within the network. It is designed to Integrity, Authentication and Non-repudiation.

4.2.1 Certification Authority

ARAN requires the use of a trusted certificate server T, whose public key is known to all valid nodes. Before entering the ad hoc network, each node must request a certificate from T. Each node receives exactly one certificate after securely authenticating their identity to T. A node A receives a certificate from T as follows:

\[ T \rightarrow A: \text{Cert}_A = [\text{IP}_A, K_A^+, t, e]K_T^- \]

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 \( e \) when the certificate expires, signed with the private key of the T.

4.2.2 Authenticated Route Discovery

When a source node wants to establish a route to a particular destination, it initiates a route discovery process. It creates a RDP (Route Discovery packet) and sends it to all its neighbors.

\[ A \rightarrow [\text{RDP}, \text{IP}_x, \text{Cert}_A, N_A, t] K_A^{-1} \]

RDP = Packet identifier.
IPx = IP Address of Destination.
Na = Nonce/Sequence number for that particular route discovery
t = Time Stamp.
KA^{-1} = Private Key of A.

When a node receives an RDP message, it sets up a reverse path back to the source by registering the neighbor from which it received the RDP. The receiving node uses A’s public key to validate the signature and verify that A’s certificate has not expired. It also checks the \((N_A, \text{IPA})\) tuple to verify that it has not already processed this RDP. 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.

\[ B \rightarrow \text{broadcast}: [[\text{RDP}, \text{IP}_x, \text{Cert}_A, N_A, t]K_A^{-1}K_B^{-} \cdot \text{Cert}_B] \]

Upon receiving the broadcast, B’s neighbor C validates the signature with the given certificate. C then removes B's signature, signs the original RDP from A with its own signature and then rebroadcasts the RDP packet to its neighbors.

\[ C \rightarrow \text{broadcast}: [[\text{RDP}, \text{IP}_x, \text{Cert}_A, N_A, t]K_A^{-1}K_C^{-}, \text{Cert}_C] \]

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 destination unicasts a Reply (REP) packet back along the reverse path to the source.

\[ X \rightarrow D: [\text{REP}, \text{IP}_A, \text{Cert}_X, N_A, t] K_X^{-} \]

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.

\[ D \rightarrow C: [[\text{REP}, \text{IP}_A, \text{Cert}_X, N_A, t]K_X^{-}]K_D^{-}, \text{Cert}_D \]

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

\[ C \rightarrow B: [[\text{REP}, \text{IP}_A, \text{Cert}_X, N_A, t]K_X^{-}]K_C^{-}, \text{Cert}_C \]

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 guaranteeing a loop free route.

4.2.3 Route Maintenance

In ARAN 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 deactivated.
in the route table. Data received on an inactive route causes node to generate an error (ERR) message that travels the reverse path to source. Error messages are also used to report a broken link due to node movement. For a route between a source A to destination X, an error message from node B to neighbor C looks like:

\[ B \rightarrow C: [\text{ERR}, \text{IP}_A, \text{IP}_X, \text{Cert}_C, \text{Nb}, t]K_B^- \]

- ERR = Packet type identifier.
- IP_A = IP address of source.
- IP_X = IP address of destination.
- Cert_c = Node that cannot be reached
- Nb = Nonce of B.
- T = Timestamp to indicate freshness.

This message is forwarded along the path toward the source without modification. It is 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. ARAN does not differentiate erratic behavior of a malicious node and malfunctioning of a friendly node. ARAN’s response to erratic behavior is a local decision and the details are left to implementors.

4.2.4 Key Revocation

ARAN also includes a method to revoke a certificate issued to a particular node, if it is determined that the node is malicious. 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:

\[ T \rightarrow \text{broadcast}: [\text{revoke}, \text{Cert}_r]K_T^- \]

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

4.3 SEAD

Secure Efficient Distance Vector Routing protocol for Ad-hoc networks is the secure extension of DSDV protocol. SEAD uses one-way hash chains to authenticate the routing updates for a particular node.

4.3.1 Assumptions

SEAD assumes that all the nodes in the network are aware of the maximum network diameter m. The network diameter essentially means that all the nodes should be aware of the maximum hop count for any given route. SEAD uses one-way hash to authenticate all the routing updates; it therefore assumes some mechanism to distribute the commitment of all nodes to all nodes in the network. It also requires that all the nodes generate a one-way hash chain of length (n) that is a multiple of the network diameter (m).

4.3.2 Basic design

The basic design of SEAD is based on DSDV (section 2.1). The design differs from DSDV by using a different mechanism to authenticate and generate routing updates. SEAD does not depend of weighted settling time to trigger an update. Weighted settling time is essentially the average time it takes to get a best routing update for a particular node. SEAD directly selects the routing update with the maximum sequence number and minimum metric.

4.3.3 Metric and sequence number authenticators

A source node generates one-way hash key chain by repeatedly applying a hash function to random value X. It then creates a chain of n elements and then discloses the commitment of the chain to all the nodes in the network. The one-way hash chain conceptually provides authentication for the lower bound of the metric in other routing updates for this destination; the authentication provides only a lower bound on the metric, since it does not prevent a malicious node from claiming the same metric as the node from which it heard this route. In particular, the one-way hash function provides the property that another node can only increase a metric in a routing update, but cannot decrease it. The method used by SEAD for authenticating an entry in a routing update uses the sequence number in that entry to determine a next group of m elements from that destination node’s hash chain, one element of which must be used to authenticate that routing update. The particular element from this group of elements that must be used to authenticate the entry is determined by the metric value being sent in that entry. Specifically, if a node’s hash chain is the sequence of values

\[ h_0, h_1, h_2, h_3 \ldots h_n \]
and \( n \) is divisible by \( m \), then for a sequence number \( i \) in some routing update entry, let \( k = n/m - i \). An element from the group of elements

\[
hkm; hkm+1; \ldots; hkm+m-1
\]

from this hash chain is used to authenticate the entry; if the metric value for this entry is \( j \), \( 0 \leq j < m \), then the value \( hkm+j \) here is used to authenticate the routing update entry for that sequence number.

When a node in SEAD sends a routing update, the node includes one hash value with each entry in that update. If the node lists an entry for itself in that update, it sets the address in that entry to its own node address, the metric to 0, the sequence number to its own next sequence number, and the hash value to the first element in the group of its own hash chain elements corresponding to that sequence number. In the example given above for sequence number \( i \), the node sets the hash value in that entry to its \( hkm \). If the node lists an entry for some other destination in that update, it sets the address in that entry to that destination node’s address, the metric and sequence number to the values for that destination in its routing table, and the hash value to the hash of the hash value received in the routing update entry from which it learned that route to that destination. This use of a hash value corresponding to the sequence number and metric in a routing update entry prevents any node from advertising a route to some destination claiming a greater sequence number than that destination’s own current sequence number, due to the one-way nature of the hash chain. Likewise, no node can advertise a route better than those for which it has received an advertisement, since the metric in an existing route cannot be decreased, and metric, to confirm that the resulting value equals the prior authentic hash value. If so, the entry is authentic and the node processes it in the routing algorithm as a normal received routing update entry; otherwise, the node ignores the received entry and does not modify its routing table based on it.

### 4.3.4 Neighbor Authentication

The source of each routing update message in SEAD must also be authenticated, since otherwise, an attacker may be able to create routing loops. A broadcast authentication protocol like TESLA may be assumed, but since this protocol assumes time synchronization the authors of this paper propose to use shared secret keys between each pair of nodes. This scheme is efficient but it is extremely expensive from memory usage point of view. SEAD includes periodic neighbor sensing functionality, each node knows the set of neighbors for which it needs to authenticate routing updates. When two nodes first become neighbors, one of the two nodes will transmit a routing update first. That update will cause the receiving node to detect the new neighbor. As a result of hearing this update, the receiving node will send a triggered routing update, allowing the other node to detect.

### 5. Security Evaluation

<table>
<thead>
<tr>
<th>Attacks</th>
<th>ARIADNE</th>
<th>ARAN</th>
<th>SEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modification</td>
<td>Yes</td>
<td>Yes</td>
<td>No/Can increase metric</td>
</tr>
<tr>
<td>Impersonation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fabrication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Packet Dropping</td>
<td>Yes/Discovered later</td>
<td>No, if node compromised</td>
<td>No</td>
</tr>
<tr>
<td>Gratuitous detour</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Eavesdropping</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DoS</td>
<td>No/Proposes a scheme</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Unauthorized Participation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 5.1 Modification

Ariadne does not allow modification of data while the packet is in transit, this feature is achieved by sharing a secret key between sender and receiver. ARAN prevents modification of data by not allowing untrusted nodes in the network. However, if a node is compromised which has a valid certificate from the server then it can modify the data, but ARAN also provides Non-repudiation. Hence a node cannot deny that changes it had made to a particular packet. Additionally it also implements the Key revocation feature. SEAD guards against modification in a way that it does not allow decrease of a metric or increase of a sequence number, a feature achieved by one-way hash-chains. However, it does allow a node to increase the metric.

#### 5.2 Fabrication
Ariadne does not allow fabrication of messages, it achieves this feature with the help TESLA authentication protocol, by requiring each node to authenticate the Error packet with TESLA key. ARAN guards against this attack by using a certification authority T. ARAN however does not guard against this attack if a trusted node having a valid certificate has been compromised. In SEAD, an attacker that has not compromised any node cannot successfully send any routing messages, since an uncompromised neighbor node will reject the messages due to the failed neighbor authentication.

5.3 Packet Dropping

Ariadne defends against black hole attacks by making use of a shared secret key between the sender and the receiver. A malicious node therefore cannot drop a packet and forge an acknowledgement for the same. Further, if a node just drops the packet without sending the acknowledgement then the sender will eventually come to know since it will not receive acknowledgement for the same. ARAN guards against this attack by not allowing unauthorized participation in routing, however if a trusted node is compromised then the security is vulnerable to this kind of attack. SEAD does not guard against this attack.

5.4 Gratuitous Detour

Ariadne requires use of TESLA keys in which a mechanism is assumed to share the commitment with the help public cryptosystem. Hence a node cannot add nodes in a route since it then has to have to commitment and public keys for all the nodes that have been added. ARAN defends against this attack by using Hop to Hop Authentication. SEAD does not defend against this attack.

5.5 DoS

Ariadne does not defend against DoS attacks, but the authors propose a scheme e.g. watchdog or pathrater to limit the damage of such attacks. ARAN does not defend against this attack, since a set of attacker nodes can send spurious routing messages and a trusted node may spend lot of time and energy in validating the requests. SEAD does not defend against this attack.

5.6 Unauthorized participation

Ariadne requires use of TESLA keys in which a mechanism is assumed to share the commitment with the help public cryptosystem. ARAN uses certification authority to guard against this kind of attack. SEAD also assumes some kind of system to share the commitment, which authenticates each and every user in the network.

6. Performance Evaluation

My initial goal when I started this project was to reproduce the results that were cited by the authors in their respective papers. Then I wanted to conduct my own set of experiments to test the protocols in different set of scenarios. Unfortunately the code that was provided by the authors didn’t work with simulator NS-2, since they had lost the original code in a hard-drive crash. I spent the entire semester installing the old version of the simulator which was not compatible with the current version g++ compilers. Finally I reached a point where I could successfully install the simulator but then could not run the partial code provided by the authors.

I therefore present the results of network performance analysis as was presented by the authors of the papers.

6.1 Tools used

GloMoSim:
Global Mobile Information Systems simulation library. GloMoSim is a scalable simulation environment for wireless and wired network systems being developed by Computer Science department at University of California, Los Angeles. This was used to simulate ARAN protocol.

Network Simulator -2
Network Simulator is being developed by University of Southern California’s Information Science Institute (ISI) targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS2 has been used to simulate SEAD and ARIADNE.

6.2 Evaluation criteria.

Given below are the list of parameters on which ARIADNE and SEAD have been evaluated. The authors of this protocol have used DSR as the base to protocol to compare the results.
Packet Delivery Ratio (PDR): The fraction of application level data packets sent that are actually received at the respective destination node.

Packet Overhead: The number of transmissions of routing packets; for example, a ROUTE REPLY sent over three hops would count as three packets in this metric.

Byte Overhead: The number of transmissions of overhead (non-data) bytes, counting each hop as above.

Mean/Average Latency: The average time elapsed from when a data packet is first sent to when it is first received at its destination.

ARAN has been simulated in GlomoSim and the authors have used AODV as the reference to evaluate the protocol on the basis of following parameters.

Packet Delivery Fraction: This is the fraction of the data packets generated by the CBR sources that are delivered to the destination. This evaluates the ability of the protocol to discover routes.

Routing Load (bytes): This is the ratio of overhead bytes to delivered data bytes. The transmission at each hop along the route was counted as one transmission in the calculation of this metric. ARAN suffers from larger control overhead due to certificates and signatures stored in packets.

Routing Load (packets): Similar to the above metric, but a ratio of control packet overhead to data packet overhead.

Average Path Length: This is the average length of the paths discovered by the protocol. It was calculated by averaging the number of hops taken by each data packet to reach the destination.

Average Route Acquisition Latency: This is the average delay between the sending of a route request/discovery packet by a source for discovering a route to a destination and the receipt of the first corresponding route reply.

6.3 Simulation environment

GloMoSim: When GloMoSim was used for simulation (ARAN) the following simulation environment was used:

<table>
<thead>
<tr>
<th>Model</th>
<th>Two-ray ground reflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac Layer</td>
<td>802.11</td>
</tr>
<tr>
<td>Traffic type</td>
<td>CBR over UDP</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>20</td>
</tr>
<tr>
<td>Terrain size</td>
<td>670m * 670m terrain</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>50</td>
</tr>
<tr>
<td>Terrain size</td>
<td>1000m * 1000m</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250m</td>
</tr>
<tr>
<td>Number of sessions</td>
<td>5</td>
</tr>
<tr>
<td>Packet Size</td>
<td>512 bytes</td>
</tr>
<tr>
<td>Node Speeds</td>
<td>0, 1, 5, 10 m/sec</td>
</tr>
<tr>
<td>Data rate</td>
<td>4 packets / sec</td>
</tr>
<tr>
<td>Packet processing delay</td>
<td>2 ms</td>
</tr>
<tr>
<td>Signature generation delay</td>
<td>8.5ms</td>
</tr>
<tr>
<td>Signature generation delay</td>
<td>0.5ms</td>
</tr>
</tbody>
</table>

ARAN Specifications used:

| Key Size               | 512 bits                  |
| Signature Size         | 16 bytes                  |

ARAN Specifications used:

| Model                  | Two-ray ground reflection |
| TeslA time interval    | 1 second                  |
| Pessimistic end-to-end propagation time (τ) | 0.2 second |
| Maximum time synchronization error (Δ) | 0.1 second |
| Hash length            | 80 bits                   |

SEAD Specifications used:

| Hash length            | 80 bits                   |

6.4 ARAN Results

The results obtained for simulating ARAN with above specifications were compared with results from simulation of AODV. The comparison of the results is shown in the graphs below.
6.4.1 Analysis

It can be inferred from the graphs that packet delivery ratio for ARAN was about 95% at all times like AODV. The high percentage of delivery ratio shows ARAN’s ability to discover and maintain routes, even with high node mobility.

Due to the addition of more data in control packets to achieve security, the byte routing load is significantly higher and increases to nearly 100% for nodes moving at 10m/s as compared to 45% for AODV. Even though the data in control packets increase, the number of control packets transmitted by the two protocols remains roughly equivalent.

Looking at the average latency, we realize that average latency for ARAN almost identical as for ARAN, but the average route acquisition latency is approximately one and a half times that for AODV. This is due to that fact that when each node processes a control packet, it has to verify the digital signature of the previous node and then replace it with its own digital signature, which takes up some additional time.
6.5 ARIADNE and SEAD results

Packet delivery Ratio (PDR)

From the graphs it is evident that Ariadne delivers fewer packets than DSR at higher mobility. The reason for the same can be attributed to slow route discovery process involving TESLA authentication and secondly, because cannot be processed until TESLA key is disclosed. SEAD outperforms DSDV in terms of PDR, since it does not wait for average settling time to trigger updates.

Byte and Packet Overhead:

In case of Ariadne the packet overhead is considerably low as compared to DSR, since ARIADNE results in more stable routes than DSR. However the Byte overhead is extremely high in the case of ARIADNE due to the authentication overhead in Route Request, Reply and Error. SEAD results in increased overhead due to the addition of hash value in each routing update.

Average Latency:

Ariadne has higher latency than DSR. The reason for this can be attributed to the time required for TESLA authentication in case of route errors.

7. Conclusion

To conclude I believe that it is difficult to suggest which of the above protocols is the best as compared to the rest. All the above described protocols have been designed to guard against the well known attacks, however some of them still leave issues untouched. The security relies heavily on having all the nodes in the network to be good. Once a single node in the network is compromised entire security of the network is at stake. Further, performance analysis of all the protocols could not be done on a single platform for e.g. ARAN was
simulated on GlomoSim and ARIADNE and SEAD were simulated on Network Simulator-2.

ARIADNE:

Ariadne is one of the most innovative of designs and extremely complex to understand. However the complex design leads to a robust and secure protocol. The only disadvantage is that ARIADNE requires time synchronization, which I believe is not a well accepted feature in real world networks. Ariadne is also memory expensive. The protocol implicitly implies that all the nodes need to share the commitment with all other nodes in the network. This basically means that all nodes should have shared key for all other nodes in the network.

ARAN:

ARAN depends on Asymmetric cryptography for security and also has high performance overhead. The authenticity of the node is dependent on the IP address. I believe this assumption is not credible since it is inappropriate to have static IP address for mobile nodes. The entire security revolves around the trusted certification server T. Compromising the certification authority compromises the entire network. It has a good feature of key revocation which could limit the extent of damage caused by a compromised node.

SEAD:

It is a distance vector ad hoc routing protocol and uses one-way hash functions for security. It authenticates neighbors before sending out any routing information or data packets and does not require nodes to have any time synchronization. It has lower packet overhead and a higher packet delivery rate even at lower pause time than ARIADNE. However, it has not been designed to handle modern attacks such as Black hole, grey hole partition of network etc. Further, it is fundamentally difficult to secure distance vector protocol since all the routing information has to be compressed into hop count value and next hop. Additionally, it is difficult to incorporate features to guard against future security attacks.

8. References


5. Adrian Perrig, Ran Canetti, J.D. Tyagar, Dawn Song, “The TESLA Broadcast Authentication Protocol” in RSA Cryptobytes , Volume 5 (Summer), 2002


8. RFC 1058 – Distance Vector Protocol

9. RFC 1583 – Optimized Link state protocol
http://www.faqs.org/rfcs/rfc1583.html


Lecture Notes in Computer Science, 2002.