25

Trust Establishment in Mobile Ad Hoc Networks: Direct Trust Distribution-Performance and Simulation

Dawoud D.S.[1], Richard L. Gordon[2],
Ashraph Suliman[1] and Kasmir Raja S.V.[3]
[1]*National University of Rwanda*
[2]*University of KwaZulu Natal*
[3]*SRM University, Chennai,*
[1]*Rwanda*
[2]*South Africa*
[3]*India*

1. Introduction

In the previous chapter, we discussed the distinct characteristics of ad hoc networks, which make them very difficult to secure. Such characteristics include: the lack of network infrastructure; no pre-existing relationships; unreliable multi-hop communication channels; resource limitation; and node mobility. We provided a theoretical background to mobile ad hoc networks and the security issues that are related to such networks. We defined the ad hoc networks and their characteristics in terms of trust establishment. As the focus of the two chapters is on the network layer, attacks specific to this layer are identified and explained in Chapter 1. We also presented a survey of the existing key management solutions for mobile ad hoc networks.

The current chapter is a continuation for the previous one. This is why we start this chapter by Section-2 that offers a survey of the existing secure routing protocols for mobile ad hoc networks. This section makes a pertinent observation that most secure routing protocols assume some kind of key management authority exists. Mobile ad hoc networks have little fixed network architecture and it is unlikely that there is a centralised authority member. Section-2 of this chapter together with last section of the previous one identify the problem that the two chapters together are addressing. There exists secure routing mechanisms to address the unique characteristics of mobile ad hoc networks, however, these solutions assume that key management is addressed prior to network establishment. A novel, on-demand solution to the key management problem for mobile ad hoc networks is then described. Section-3 details the functionality and operation of the proposed model: "Direct Indirect Trust Distribution" (DITD). The DITD model focuses on the task of distributing keying information. The DITD model also includes a verification optimization protocol and trust evaluation metric, which maximises the security of distribution.

The implementation and simulation of the DITD model is examined in Section-4. There are various packages used to compare existing and proposed routing protocols. One such

package is the ns2 Network Simulator, which is commonly used in the relating literature. A comparative ns2 simulation study between the DITD and the AODV protocols is presented. The DITD model is based on the Ad Hoc On-demand Distance Vector (AODV) routing protocol. Simulations show the performance overhead of including key management functionality and the performance of DITD in the presence of malicious attacking nodes.
Section-5 summarises the contribution of the Chapter to the field of trust establishment in mobile ad hoc networks. Section-5 also provides future direction for research.

2. Secure routing in mobile ad hoc networks

A mobile ad hoc network's routing protocol has unique challenges due to the dynamic nature of ad hoc network. Mobile ad hoc networks do not have the same privileges that fixed, wired networks have. The routing mechanisms are uniquely designed to deal with the lack of infrastructure and unreliable wireless multi-hop communication channels.
This section investigates the procedure of securing of these routing protocols. The routing solutions are briefly visited and an extensive survey is presented for the existing security mechanisms that are used to secure these routing protocols.
Routing in mobile ad hoc networks is divided into two categories: table driven methods and on-demand methods. Table driven methods are also known as proactive routing. They maintain routing tables that contain routes to all the nodes in the network. Theses tables are periodically updated which allows routing information to be available at all times. Examples of table methods include Destination Sequence Distance Vector Routing (DSDV) [Perkins & Bhagwat, 1994] and Optimized Link State Routing (OLSR). Source initiated on-demand routing methods establishes routes in a reactive manner. Routes are established through a route discovery phase. During a route discovery phase node *S* will broadcast a request message *RREQ* into the network. This request message is forwarded until it reaches its target destination node *D*. Node *D* then replies with a reply message *RREP* which is unicast along the reverse route, until it reaches the source and the route is established. Routes are maintained as long as they are required. Examples of on-demand methods include Ad Hoc On-Demand Distance Vector (AODV) [Perkins et al, 2003] and Dynamic Source Routing protocol (DSR) [Johnson et al, 2001]. The reactive on-demand approach is less computationally expensive, in comparison with the proactive table driven approach.
In the previous chapter, it is identified that most security attacks target the network layer, and more specifically the routing protocol. These attacks include: black-hole attacks; wormhole attacks; eavesdropping attacks; byzantine attacks; resource consumption attacks; and routing table poisoning. The routing protocol is found on the network layer and is a significant service for mobile ad hoc network. Adversaries, specifically, target the routing protocol. Thus, a secure routing solution is needed for ad hoc networks to be securely implemented.
This section gives a survey and an analysis of the existing secure routing protocols. Each protocol is presented and investigated based on: functionality; operational assumptions; and security. A summary and discussion is formulated at the close of this section.

2.1 Secure Efficient Ad hoc Distance vector routing protocol (SEAD)

Secure efficient ad hoc distance vector (SEAD) [Hu et al, 2002] is a secure routing protocol which is used in conjunction with the table driven destination-sequenced distance vector (DSDV) routing protocol [Perkins & Bhagwat, 1994]. The DSDV routing protocol uses a

distributed version of the Bellman-Ford algorithm to discover the shortest path between two nodes. The SEAD protocol uses symmetric key cryptography and one-way hash functions to protect against security attacks like denial of service and resource attacks.

a. System Overview

The DSDV routing protocol discovers the shortest path based on a route's hop count. Routing packets are assigned sequence numbers to ensure the most recent route is processed. The hop count and sequence number variables are stored in the routing packets. Attackers can create an incorrect routing state in nodes resulting in a denial of service attack (DoS) where the attacker attempts to make other nodes consume excess bandwidth and processing time. SEAD makes the routing process robust against multiple uncoordinated attackers by authenticating the hop count and sequence number of routing packets with a one-way hash function h. Hash chaining is used so that only nodes that are in possession of the previous routing update (identified by a sequence number) can broadcast a new routing update. Authenticated routing updates are computed to prevent against malicious routing updates broadcast by attackers.

b. One-Way Hash Function

SEAD uses a one-way hash function to authenticate routing updates and minimize resource consumption attacks. A formal definition of the hash function H is provided in [Stalling, 2003]. The most commonly used hash functions are MD-5 [Rivest, 1992] and SHA-1 [Publications F IPS, 2008].

A one-way hash function H is used to generate a one-way hash chain h. The one-way hash function H has an input of any bit length * and outputs a variable of fixed bit length p. The one-way hash function H must be computationally impossible to invert.

$$H: (0,1)^* \rightarrow (0,1)^p$$

A hash chain h_i is created when a node selects a random number $x \in (0,1)^p$ and uses it to generate a list of variable which make up a hash chain $h_0, h_1, h_2, h_3, \ldots, h_n$. Here $h_0 = x$ and h_i is calculated using the irreversible one-way hash function H such that:

$$h_i = H(h_{i-1}) \text{ where } 0 \leq i \leq n$$

Assuming there is an existing authenticated element, a node can verify elements later in the chain's sequence. For example if an authenticated element h_i exists, a node can authenticate h_{i-4} by checking that $h_i = H(H(H(H(h_{i-4}))))$. SEAD assumes the existence of an authentication and key distribution mechanism to distribute an authenticated element like h_n allowing for authentication by hash chaining [Hu et al, 2002].

c. Authenticating routing updates

SEAD uses the elements of the hash chain to provide authentication and secure the routing updates in DSDV. SEAD assumes an upper bound on the variable to be authenticated, for example if it were the hop count then SEAD would assume a maximum route distance n in the network (the maximum hop count between two nodes allowed). This also eliminates any routes with a length greater than m to exist, eliminating possible routing loops or the routing infinite problem.

The sequence values that make up the hash chain are calculated from the H function such that $h_1, h_2, \ldots, h_n$ where n is divisible by m. For a routing table entry with sequence number i

let $k = {}^{n}/_{m} - i$. An element from h_{km}, h_{km+1}, …, h_{km+m-1} is used to authenticate the routing entry with sequence number i. If the hop count is j where $0 \leq j < m$, then h_{km+j} is used to authenticate the routing entry found with sequence number i and hop count j [Hu et al, 2002].

Routing updates are sent with the appropriate routing information and a hash chain value is used to authenticate the update. If the authentication value appended is h_{km+j} then only attackers with h_{km+j-1} can modify the authentication value. Nodes receiving a routing update, check the authentication value h_{km+j} by calculating the new hash chain value. Receiving nodes can calculate the new hash chain value by using the earlier hash chain value h_{km+j-1} and the received sequence number i and hop count j. If the new calculated hash value is equal to h_{km+j} then the routing update is verified.

SEAD proposes two methods for routing update authentication. One method uses clock synchronization and a broadcast authentication mechanism like TESLA [Perrig et al, 2001]. The second method requires a shared secret between each communicating node pair. The secret can be used to implement a message authentication code (MAC) between nodes authenticating routing update messages.

d. Analysis

The SEAD protocol protects the ad hoc network from routing attacks that target resource consumption. The SEAD protocol does protect against multiple uncooperative attacks, preventing routing loops but routing loop prevention cannot be guaranteed in the presence of co-operating attackers. The SEAD protocol is vulnerable to intelligent attackers that use the same sequence number and same hop count of the most recent update to corrupt routing information. The SEAD protocol provides protection against denial of service attacks [Perrig et al, 2001], replay attacks and routing table poisoning by authenticating routing updates so malicious nodes cannot corrupt the routing procedure.

2.2 A secure on-demand routing protocol for ad hoc networks (Ariadne)

Ariadne [Hu et al, 2005] is a secure on-demand routing protocol which uses symmetric cryptography. Ariadne is based on the on-demand DSR [Johnson et al, 2001] routing protocol and is developed by the same authors as the SEAD protocol [Hu et al, 2002]. Ariadne provides end-to-end authentication on the routing layer.

a. System Overview

Ariadne assumes a shared secret key between communicating node pairs and uses message authentication code (MAC) to authenticate end-to-end packets between the communication pair. Broadcast authentication is employed, with loose time synchronization, to authenticate route request and other broadcast packets. The TESLA [Perrig et al, 2001] broadcast authentication scheme is used. In TESLA the source generates a one-way key chain and a schedule is made which defines at which time keys of the chain are revealed. This mechanism limits Ardiadne's operation to ad hoc networks which have time synchronization. Ardiane provides end-to-end authentication in an on-demand manner over the DSR routing protocol [Hu et al, 2005].

b. End-to-end Authentication

For communication from a source node S to a destination node D, the source S will broadcast a route request into the network and expect a reply from D. Ariadne assumes a

shared secret between *S* and *D*, K_{SD} and K_{DS}, which enables message authentication for each respective direction.

Nodes *S* wanting to start a route discovery for node *D* will first generate an initial hash chain h_0 consisting of: a packet identifier identifying the type of packet (a request packet *RREQ* in this case); the source's address (ID_S); the destinations address (ID_D); a broadcast identity (*bi*) identifying the current route discovery; and a TESLA time interval (*tes*) identifying the expected time of arrival at the destination.

$$h_0 = MAC_{K_{SD}}(RREQ|ID_S|ID_D|bi|tes)$$

Node *S* will broadcast a route request packet which includes: a packet identifier, the hash chain h_0; the source's address (ID_S); the destinations address (ID_D); the broadcast identity (*bi*); the TESLA time interval (*tes*); a node list *N()* and a MAC list *M()*. The packet broadcast is as follows:

$$S \rightarrow broadcast \ : \ RREQ|h_0|ID_S|ID_D|bi|tes|N()|M()$$

A neighbouring node that receives the route request checks the validity of the TESLA time interval, *tes*. The TESLA time interval is valid if the corresponding key that it points to has not been revealed yet and the time interval does not point too far in the future. The neighbouring node *A* will then compute a new hash chain h_1 using the previous hash chain h_0. A message authentication code of the packet to be broadcast is created (MAC_A). MAC_A is calculated using the TESLA key (K_{Ates}). Before forwarding the packet the neighbour node *A* includes: the hash chain h_1; itself in the node list *N*; and the MAC_A calculated in the MAC list *M*. The hash function and broadcast packet are as follows:

$$h_1 = H(A|h_0)$$

$$A \rightarrow broadcast \ : \ RREQ|h_1|ID_S|ID_D|bi|tes|N(A)|M(MAC_A)$$

Intermediate node *P* receiving a forwarded route request first calculates a new message authentication code MAC_P and a new hash chain $h_i = H(P-1|h_{i-1})$ where *P*-1 is the previous node and h_{i-1} is the previous hash chain value. Secondly it includes this information and forwards the route request as follows:

$$P \rightarrow broadcast \ : \ RREQ|h_i|ID_S|ID_D|bi|tes|N(A,\dots,P)|M(MAC_A,\dots,MAC_P)$$

The route request is propagated to the destination node *D*. When *D* receives the route request it validates the authenticity of the route request by checking that the TESLA time intervals indicate no keys have been released as of yet and that the hash chain is valid. *D* then generates a message authentication code MAC_D. MAC_D and an empty key list *K()* are included in the packet and sent back along the reverse path indicated by the node list and DSR protocol. The MAC_D and reply message are as follows:

$$MAC_D = MAC_{K_{DS}}(RREP|ID_D|ID_S|bi|tes|N(\dots)|,M(\dots)$$

$$D \rightarrow P \ : \ RREP|ID_D|ID_S|bi|tes|N(\dots)|M(\dots)|MAC_D|K()$$

Intermediate node that receive a reply message will wait for the *tes* time interval to lapse so the corresponding key can be revealed an included in the key list *K()*. The reply message is forwarded until it contains all the TESLA keys of the intermediate nodes and it finally

reaches the source node *S*. The source then verifies the validity of all the keys, MAC_D, and the message authentication code contains.

c. Maintenance

Ariadne achieves secure route maintenance by authenticating the DSR error messages. Ariadne authenticates error messages preventing malicious nodes from broadcasting false broken links and causing denial of service type attacks. When an error message is generated TESLA authentication information is included. If authentication is delayed as a result of the TESLA time intervals, the intermediate nodes buffer the error message until the appropriate keys are revealed and the message can be authenticated and action taken [Hu et al, 2005].

d. Analysis

The authors of Ariadne are the same authors of SEAD [Hu et al, 2002] protocol. Ariadne employs an end-to-end approach to authentication while SEAD uses a hop-by-hop approach because of the DSDV routing procedure. The Ariadne proposal is based on the on-demand DSR routing protocol. Ariadne implements TESLA broadcast authentication and message authentication code to provide authentication for routing packets in an ad hoc network environment. The Ariadne proposal assumes that there exists some shared secret between a communication pair, therefore assuming the existence of an authentication and key distribution mechanism. Ariadne relies on TESLA authentication which requires time synchronization in the ad hoc network, synchronization is difficult to achieve without the presence of an outside authorized member or TTP.
Ariadne implements end-to-end authentication to prevent unauthorized nodes from sending error messages and incorrect routing packets in the form of repays attacks. However this proposal does not consider the case where attackers do not cooperate with the routing protocol and drop routing packets which are suppose to be forwarded. An extension is proposed in [Hu et al, 2003] which uses packet leashing to solve this problem.

2.3 Authenticated Routing for Ad hoc Networks

The authenticated routing for ad hoc networks (ARAN) protocol [Sanzgiri et al, 2002] is a securing routing solution which uses cryptographic certificates. ARAN is designed for an on-demand ad hoc routing protocol and achieves authentication, integrity and non-repudiation on the network layer but assumes prior shared secrets at initialization.

a. System Overview

The ARAN secure routing protocol establishes trust in three stages:
1. Issuing of certificates
2. Route Discover process
3. Shortest path Optimization

Initially ARAN assumes the presence of a trusted third party (TTP) which issues valid certificates, and a shared public key for all participating nodes. The route discovery process of ARAN provides end-to-end authentication for communicating nodes. The source node broadcasts a route request which carries the source's certificate. The route request is propagated to the destination node by an end-to-end authentication process. The destination node responds by unicasting a reply message back along the found route using the end–to-end authentication protocol.

b. Issuing of Certificates

This section describes how the certificates are issued and distributed to the participating nodes. The assumption is made that an authenticated trusted third party (TTP) member exists which plays the roles of an initial certificate authority (CA). This TTP CA is known to all the nodes in the network. The ARAN protocol assumes that certificates are generated by the TTP CA and distributed to nodes before they officially join the wireless ad hoc network. No specific key distribution mechanism is described for the ARAN protocol. Node *i* entering the network will receive a certificate $cert_i$ from the TTP CA that has the following contents:

$$TTP - CA \rightarrow i \;:\; cert_i = E_{k_{TTP-CA}}(ID_i|K_i|t|et)$$

The certificate $cert_i$ is signed by the private key of the TTP-CA ($k_{CA\text{-}TTP}$) and has the following contents: ID_i representing the identification of node *i* for example a specific IP address; K_i the public key of node *i*; *t* the timestamp for the $cert_i$; and *et* the expiry time of the certificate.

c. Route Discovery Process

The route discovery process provides end-to-end authentication which ensures that the packets sent from a source node *S* reach their intended destination node *D*. Each node maintains a routing table which contains the active communication routes between the different source and destination pairs. The route discovery process begins by a source *S* broadcasting a route request. The route request is signed by the source node's private key k_S and contains: the certificate of the source node ($cert_S$); the identification of the destination node (ID_D); a nonce (N_S); a timestamp (*t*); and a packet identifier identifying that the packet is a route request packet (*RREQ*). The authenticated route request broadcast by node *S* is:

$$S \rightarrow broadcast \;:\; E_{k_S}(cert_S|ID_D|N_S|t|RREQ)$$

The nonce value is incremented every time the source sends a route request. The nonce value acts like a sequence number ensuring the most recent route request is dealt with. Each node that receives the route request will process it if it has a higher value of the source's nonce than previously received route requests from the same source node. Each intermediate node *P* receiving the route request will validate the signature with the certificate, update the routing table with the neighbour from whom it received the route request, sign the route request and broadcast it to its neighbours. Node *P* will remove the signature and certificate of the previous node if the previous node was not the source itself. Therefore each forwarded route request is authenticated by the source and the intermediated node and will contain two certificates $cert_S$ and $cert_P$:

$$P \rightarrow broadcast \;:\; cert_P|E_{k_P}(E_{k_S}(cert_S|ID_D|N_S|t|RREQ)$$

The route request is propagated to the destination node *D* which will reply with a reply message *RREP*. The reply packet is signed by the destination node's private key k_D and the packet contains: the identity of the source node (IDs); the destination's certificate ($cert_D$); a nonce of validity (N_D); a timestamp (*t*); and a packet identifier (*RREP*). The reply packet is unicast along the reverse path toward the source node with a similar authentication procedure to the forwarding of the route request.

$$D \rightarrow reverse\ path \;:\; E_{k_D}(cert_D|ID_S|N_D|t|RREP)$$

$$P \rightarrow reverse\ path \ : \ cert_P|E_{k_P}(E_{k_D}(cert_D|ID_S|N_D|t|RREP)$$

The source node will receive the reply packet *RREP* and check the signature and nonce (N_D) to verify that the packet was sent by the destination node and not a malicious attacker. If the nonce or certificate fails an error message is broadcast and the route request process restarted.

d. Shortest Path Confirmation

This is an optional procedure employed by ARAN to ensure that the shortest path is found between source and destination. Path confirmation has a high computational cost. After a route has been found between *S* and *D* the shortest path confirmation process begins. The source will broadcast a packet signed by the public key of *D* (K_D) containing: the certificate of the source; the identity of the destination node; a nonce (N_S); timestamp (*t*); and packet identifier identifying that this is a shortest path confirmation packet (*SPC*).

$$S \rightarrow braodcast \ : \ E_{K_D}(cert_S|ID_D|N_S|t|SPC)$$

Each intermediate node that receive the *SPC* packet updates its routing table, signs the packet, includes its certificate and signs it with the public key of the destination node.

$$P_1 \rightarrow braodcast \ : \ E_{K_D}(cert_{P_1}|E_{k_{P_1}}(E_{K_D}\left(cert_S|ID_D|N_S|t|SPC\right)$$

The destination node will verify all the signatures and reply to the first and subsequent *SPC* packets with a recorded shortest path packet *RSP*. The *RSP* is propagated to the source which confirms the shortest path by verifying the nonce N_S sent with the *SPC* packet.

e. Maintenance

The ARAN solution uses error messages and implicit revocation of certificates to maintain routes. Error message packets (*ERR*) are broadcast by any node *P* that discovers a broken route. An *ERR* packet is signed by its originator and includes the certificate of the originator, the source and destination pair describing the broken route, a nonce, a timestamp, and a packet identifier. Each node receiving an ERR packet will check its routing table if it contains the accused route. If it does then the *ERR* packet is rebroadcast unchanged.

$$P \rightarrow braodcast \ : \ E_{k_P}(cert_P|ID_S|ID_D|N_P|t|ERR)$$

The expiration (*et*) attribute included in each certificate allows for implicit revocation of certificates. Certificates are implicitly checked during the route discovery process. Explicit revocation is achieved by the TTP CA broadcasting a certificate revocation message to nodes which then can forward it. Routes are re-calculated as a result of certificate revocation.

Analysis

The ARAN solution uses asymmetric key cryptography to provide authentication, integrity and non-reputation. Asymmetric cryptography will result in high complexity and computational cost. A trusted certificate authority (TTP CA) is required so that authentication can be made available. In the route discovery process unlike AODV, ARAN disallows intermediate nodes which have a path to the destination to reply with a *RREP* message. This creates addition routing overheads but ensures authentication [Sanzgiri et al, 2002].

2.4 Secure Ad hoc On-demand Distance Vector (SAODV)

Zapata et al [Zapata, 2002] proposes the Secure Ad hoc On-demand Distance Vector (SAODV) protocol as a security extension to the AODV protocol. SAODV secures the AODV protocol by using a hybrid cryptographic approach involving asymmetric cryptography in the form of digital signatures and symmetric cryptography in the form of hash chains.

a. System Overview

SAODV defines the fields in a routing packet into two categories mutable and non-mutable. The non-mutable fields are authenticated using asymmetric cryptographic signatures. The only mutable field in an AODV routing packet is the hop count. A new hash chain is created for each route discovery phase which is used to secure the hop count. SAODV requires that the AODV routing packet is extended to include the security information like digital certificates. The implementation of digital signatures and hash chains provides end-to-end authentication for the AODV routing protocol.

SAODV uses asymmetric cryptography and assumes the presence of a key management scheme to distribute keys in a secure manner [Zapata, 2002]. It also assumes that it is possible to verify the relationship between a public key and an IP address or identity.

b. Packet Extension

SAODV proposes an extension to the standard AODV message format so that security mechanism can be implemented. The SAODV extension contains the following fields as described in Table - 1 and Figure 1 [Zapata, 2002].

The standard AODV protocol uses packet sizes of 512 bytes but the SAODV extension requires the AODV packet size to be extended to use packets of size 1024 bytes.

c. Route Discovery Process

A source node *S* initiates a route request to a destination node *D* by performing the following steps:

Field	Description
Type	This is a packet identifier where the value 64 identifies a request packet *RREQ* and the value 65 identifies a reply packet *RREP*.
Length	The length of the packet data.
Hash Function	This describes the hash function used for example MD5 [Rivest, 1992] or SHA-1 [Publications F IPS, 208].
Max Hop Count	The maximum hop count (*mhc*) is used for hop count authentication. It defines the maximum number of nodes a packet is allowed to pass through.
Top Hash	The top hash is the result of the hash function *H* applied *mhc* times to a random generated number *x* such that: $top\ hash = H^{mhc}(x)$. Top Hash is vital in the hop count authentication process.
Signature	This field is 32-bit aligned and contains the signature used to authenticate all the non-mutable fields in the AODV packet.
Hash	This field is 32-bit aligned and contains the hash value h_i used to authenticate the mutable hop count variable.

Table 1. RREQ and RREP Signature Extension Fields

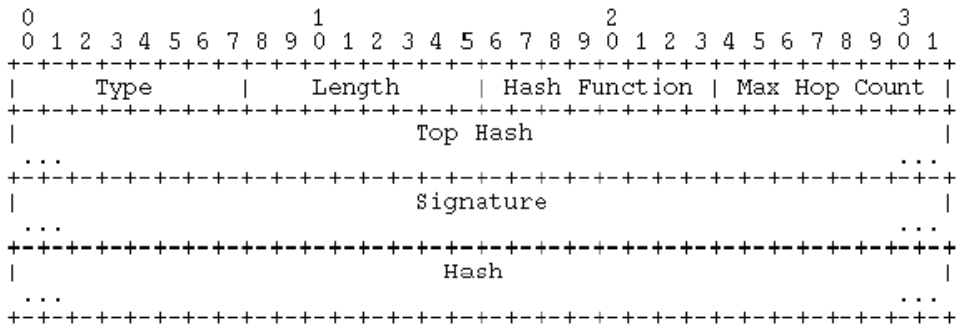

Fig. 1. RREQ Single Signature Extension

1. *S* sets the max hop count (*mhc*) variable equal to the *TTL* (time to live) variable found in the IP header.
2. *S* generates a random number *x* and sets it as the value in the hash field such that $h_0 = x$.
3. The top hash is then generated by applying the hash function *H*, max hop count (*mdc*) times to h_0 such that: *top hash*= $h_{mhc} = H^{mhc}(h_0)$. The hash function *H* is defined in the hash function field in the packet header.
4. *S* digitally signs all the fields in the packet except hop count and hash field and stores the digital signature in the 32-bit signature field.
5. *S* then broadcasts the route request packet to its neighbours.

When an intermediate node receives a route request it verifies the authenticity of the hop count and the integrity of the digital signature. The digital signature is verified using asymmetric cryptography. The hop count is verified by checking $h_{mhc} = H^{mhc-i}(h_i)$, where h_{mhc} is the top hash; h_i is the hash field of the route request; and H^{mhc-i} is the application of the hash function max hop count minus the hop count (*i*) times. The one-way hash chain approach used to authenticate the hop-count is similar to the approach used in the SEAD protocol [Hu et al, 2002]. After the intermediate node verifies the digital signature and hop count, it replaces the packet's hash field with a new hash value computed by applying the hash function to the existing hash value. The intermediate node then rebroadcasts the route request and propagates it until it reaches the destination *D*.

When the route request *RREQ* reaches the destination *D*, *D* checks the validity of the packet and will reply with a reply packet *RREP* if the route request is valid. The *RREP* packet is forward along the reverse route to the source following the same authentication and integrity procedure that the *RREQ* message experienced.

AODV allows for intermediate nodes to also reply to route requests if they have a valid route to the destination node. SAODV proposes two solutions to this security problem. The first is the simplest disallowing intermediate nodes to reply ensuring that the destination node sends the reply message *RREP* and guaranteeing authentication. The second approach uses a double signature extension which allows an intermediate node *P* to reply to a route request from *S* for *D* $RREQ_{SD}$. Intermediate node *P* will reply with a double signature *RREP* message. One signature will sign the intermediate node *P*'s standard reply and the second signature will sign the original *RREP* packet received by *P* for its route to *D*. Both reply message headers are included and sent to the source *S* to establish a secure route to *D*.

d. Maintenance

AODV uses error messages *RERR* to report broken links. SAODV secures these messages using digital signatures. The originator of the error message signs the entire message except the destination sequence number. Each forwarding node also signs the message to prevent unauthorized error messages being broadcast. Nodes using SAODV do not change their destination sequence number after receiving an error message because the error message's sequence number is not authenticated.

Analysis

SAODV authenticates the AODV routing packets preventing certain impersonation attacks. The assumption is made that a node's identity and address can be securely bound to a public key. Such an assumption leaves SAODV vulnerable to Sybil attacks.
SAODV employs asymmetric cryptography which is computationally taxing. The packet size has to be extended to incorporate the security mechanism resulting in a serve communication overhead. For every route discovery a new one-way hash chain is computed resulting in further computational overheads.
The SAODV protocol assumes a key management entity is available in the ad hoc network which can successfully distribute public keys among participants. Such infrastructure is difficult to execute in mobile ad hoc networks and before SAODV is to be implemented either an off-line TTP or distributive key management scheme must be employed.
The SAODV solution uses hash chaining and digital signatures providing security against impersonation routing attacks. It also helps toward preventing denial of service attacks and eavesdropping attacks where malicious users may re-direct a route through a malicious node where eavesdropping may occur.

2.5 Secure Link-State routing (SLSP)

A hybrid scheme is proposed in [Perrig et al, 2001] called Secure Link State Routing Protocol (SLSP). It is a proactive security solution which uses digital signatures and one-way hash chains to provide security to link-state updates and neighbour discovery. SLSP secures link state information in localized manner and can operate alone or be combined with a reactive ad hoc protocol routing protocol like ZRP where SLSP would be the intra-zone routing protocol for the Zone Routing Protocol (ZRP) [Haas & Pearlman, 2001].

a. System Overview

SLSP provides secure neighbour discovery for nodes in a limited hop radius called a zone. Link state updates are authenticated using a hybrid cryptographic method and flooding attacks prevented by a priority ranking mechanism. The main assumption of SLSP is that nodes have existing asymmetric key pairs. SLSP assumes that a key management scheme is present to certify the public keys in the network.
SLSP uses four components: key distribution, secure neighbour discovery, link state update management, and a priority ranking scheme. SLSP's priority ranking scheme prevents denial of service type attacks.

b. Key distribution

SLSP assumes that each node has an existing signed public key before it joins the network; and the certification of keys is performed by an assumed key management method. Public keys are bound to the IP addresses of the network nodes. Nodes then broadcast their public

keys into their neighbourhood zone, for example a two hop radius. The received public keys are used to authenticate future packets from the source node.

c. Secure Neighbour Discovery

SLSP uses a Neighbour Location Protocol (NLP) to proactively check that neighbouring nodes do not perform impersonation attacks. Link state information is periodically broadcast by nodes in the form of a NLP hello message. These messages are signed by the source and contain the source's MAC address (a unique hardware address) and IP address (a distinctive network address). A NLP hello message broadcast by source S is described here:

$$S \rightarrow broadcast \ : \ E_{k_S}(IP_S|MAC_S)$$

Notification messages are generated when conflicting link state information is broadcast. An inconsistent mapping of the IP and MAC addresses is considered as conflicting link state information. For example when two nodes with different IP addresses have the same physical MAC address.

d. Link State Update Management

Link state update packets are periodically broadcast to a limited hop radius of nodes. A link state update packet (LSU) contains the IP address of the source, a 32-bit sequence number used for updating and a hop count variable. The LSU hop count variable is authenticated using hash chains as discussed in SEAD [Hu et al, 2002] and SAODV [Zapata, 2002] and the rest of the packet content is authenticated using digital signatures.
When a LSU is received the digital signature is verified using the previously distributed public key. The hash chain is verified and the time to live (TTL) variable is decremented. The hop count authentication protects the LSU packet from travelling too many hops or from link state updates not to be received by nodes.

e. Priority Ranking Scheme

SLSP uses a lightweight flooding prevention scheme which gives priority ranking to nodes. A priority list is maintained for neighbouring nodes which ranks node's priority based on the number of link state updates a node broadcast. Malicious nodes will flood the network with link state update packets to cause resource and denial of service attacks. SLSP gives high priority to nodes that send less link state updates limiting the affects of flooding attacks.

Analysis

The Secure Link State Routing Protocol is a hybrid cryptographic scheme using digital signatures to provide authentication for its NLP hello messages and link state update (LSU) packets. Hash chains are used to authenticate the limited hop broadcast of LSU. Impersonation type attacks are prevented by monitoring the IP and MAC address bindings of neighbours through a neighbour location protocol (NLP). Link state updates are authenticated using digital signatures and hash chains. Flooding type attacks are prevented using priority ranking.
The SLSP protocol provides security to topology discovery but cannot act as a standalone security mechanism as it lacks a data transmission protection agent. Nodes that securely join the network can misbehave during data transmission without being detected or prevented.

SLSP lacks a disciplining agent like a revocation mechanism. For example a malicious node *B* can impersonate another node *A* and flood *A*'s neighbours with LSU packets. SLSP's priority mechanism will limit the effectiveness of the flooding attack. The NLP protocol will detect the impersonation attack but node *A* has no mechanism to correct to the attack and *A* will remain with a low priority.

2.6 On-Demand Secure routing Byzantine Resilient routing protocol (ODSBR)

The on-demand secure routing byzantine resilient routing protocol (ODSBR) is proposed in [Hu et al, 2003b]. Byzantine behaviour is defined by the authors as any action taken by an authenticated node to disrupt the routing procedure. ODSBR is a secure reputation based routing protocol that prevents the effects of byzantine failures on successful routing.

a. System Overview

ODSBR uses weighted paths to select routes in the route discovery process. Paths are assigned weights based on a fault path detection method. A high weight is assigned to an unreliable path. ODSBR is divided into three components: route discovery process, fault detection, and weight path management. ODSBR assumes that a public key infrastructure is present to manage public key authentication.

b. Route Discovery Process

Routes are discovered in an on-demand manner like in DSR. ODSBR extends the standard route request *RREQ* by adding a weight list instead of a node list like in DSR. A weight list includes the list of chained nodes with their associated weights. These weights are defined by link failures detection mechanism. The *RREQ* is signed by the source and broadcast into the network updating its weighted list after each hop until it reaches the destination. The destination then verifies the signature and broadcasts the reply message *RREP*. Each node that receives a *RREP* will then calculate the total weight of the path by summing the weights of the specific path to the current node. The *RREP* forwarded if the total weight is less than the total weight of any previously forwarded *RREP*. Before an intermediate node *P* forwards a suitable *RREP* all the signatures are verified and node *P* appends its signature. The source node receives the *RREP* and calculates the total weight and verifies all the signatures.

c. Fault Detection Method

Each node *i* has a list of probe nodes. Each probe node sends node *i* an acknowledgement message for each data packet *i* sends. If a threshold *t* of acknowledgements is not received a fault accusation is logged against a specific path. Using a binary search technique ODSBR identifies a path as faulty after *log(n)* fault accusations, where *n* is the length of the accused path.

d. Weight Path Management

A low weight is associated with a secure path. The weights associated to paths are influenced by two factors: time and fault detection. When ODSBR identifies a path as faulty based on the fault detection method then the weight for that path is doubled. Path weights are halved after a counter reaches zero, each path has an associated counter.

e. Analysis

The on-demand secure routing protocol (ODSBR) provides ad hoc on-demand routing with byzantine failure prevention. Weights are assigned to paths by a fault detection method and

paths are selected based on the weights. The Secure Routing Protocol (SRP) proposed in [Papadimitratos & Hass, 2002] introduces the metric specific path selection method but this proposal is not a standalone secure routing protocol like ODSBR [Awerbuch et al, 2008].
ODSBR assumes the existence of a public key management system. ODSBR further assumes that a shared key exists between source and destination nodes to ensure authenticity and integrity of acknowledgement messages sent by probe nodes. This helps avoid expensive asymmetric per packet computations for acknowledgement messages.
The route discovery process of ODSBR broadcast reply messages instead of unicasting them which results in a computationally expensive operation. This method will result in $2^{\frac{n}{2}+1} - 1$ reply packet transmissions where n+2 is the number of nodes in a path from node A to B including nodes A and B [Awerbuch et al, 2008]. Furthermore the cost monitoring of data packet transmission is computationally high because the fault detection method requires a threshold of t probe nodes to reply with an acknowledgement for every data packet sent.
ODSBR is identified by authors [Awerbuch et al, 2008] to be vulnerable to wormhole attacks. The wormhole attack may be avoided in the case where the wormhole link node exercises byzantine behaviour.

2.7 Reputation based CONFIDANT

The CONFIDANT protocol representing the 'Cooperation Of Nodes: Fairness In Dynamic Ad Hoc Networks' [Buchegger & Boudec, 2002] is a reputation based solution which operates over the DSR routing protocol.

a. System Overview

The CONFIDANT solution does not use any cryptographic techniques to achieve secure routing. The system is solely reputation-based and operates in an on-demand ad hoc network environment as an extension of the DSR routing protocol. Each node in the ad hoc network is required to be involved in the four components of CONFIDANT: monitoring, trust management, reputation system and path management.

b. Monitoring

Each node is a monitor and is responsible for the packets that it sends or forwards. For every packet that a node forwards it watches that the next hop node forwards the packet properly. The monitor looks for inconsistent behaviour and triggers an alarm to the reputation system if misbehaviour is discovered.

c. Trust Management

The trust management system manages the alarm messages. The alarm messages are generated by each node's monitoring system and exchanged between other nodes as to build and maintain a local rating list. The trust management manages the input of alarm messages and assigns more influence to alarm messages that come from trusted nodes and less influence from other nodes. CONFIDANT assumes pre-existing relationships between a selection of nodes called friends [Buchegger & Boudec, 2002], friend nodes are highly trusted nodes. Local rating lists are exchanged as well and their influence managed by the trust management system.

d. Reputation System

The reputation system manages and maintains the local rating list. This list contains node identities and corresponding rating. A rating will correspond to the amount of

misbehaviour a node has displayed. The ratings will be updated from alarm messages and direct observations.

e. Path Management

Paths are selected based on a rating threshold, local rating lists and a blacklist containing all untrusted nodes. A node is blacklisted when its rating is below the rating threshold *t*. The path manager removes the paths in the network which contain the blacklisted node. The path manager manages the route discovery process by reacting to route requests from blacklisted nodes or route requests that have passed through a blacklisted node.

f. Analysis

CONFIDANT is an exclusively reputation based routing protocol. Local rating lists of node's behaviour is recorded and used during the route discovery process. The authors note that only negative evidence is gathered against nodes so nodes can only be identified as less trusted as the network continues. Like most reputation based schemes a counter system is employed where each rating list entry has an associated counter. When the counter reaches zero the rating is reset to the default state of null misbehaving accusations. The CONFIDANT protocol assumes the existence of prior trust relationships between a selected number of nodes called friends [Buchegger & Boudec, 2002].

2.8 Discussion

Several different secure routing protocols are presented in Section- 2.2 they differ in the areas of security and operational requirements. The diversity of their design makes it difficult to compare their success but this section outlines the diverse characteristics of the presented protocols.

a. Security Analysis

The proposals can be categorized by the security techniques which include the asymmetric, symmetric and hybrid cryptographic security approaches. The last category is the reputation based solutions. The security mechanism of each protocol is presented and the attacks which the protocol protects against are highlighted. A summary of the evaluation is presented in Table -2.

Symmetric Cryptography

The symmetric cryptographic approaches include the Secure Efficient Ad Hoc Distance Vector Routing protocol (SEAD) and the Ariadne protocol. Hash functions and hash chains like SHA-1 [Publications F IPS, 2008] and MD5 [Rivest, 1992] are used for authentication purposes usually for the hop count variable. The hash function is lightweight compared to asymmetric security techniques.

The SEAD approach uses a hop-by-hop authentication technique. SEAD authenticates the sequence number and hop count of routing packets protecting the routing procedure from resource consumption attacks for example denial of service attacks, route table poisoning, and replay attacks.

The Ariadne protocol is proposed by the same authors of SEAD. Ariadne uses message authentication code (MAC) to provide end-to-end authentication between communication nodes. Ariadne protects against similar attacks to SEAD but uses end-to-end authentication.

Protocol	Security Approach	Techniques	Attack Prevention
SEAD [Hu et al, 2002]	Symmetric Cryptography	• Hop-by-hop authentication • Hash chains	• Resource consumption • Denial of Service • Route table attack • Replay
Ariadne [Hu et al, 2005]	Symmetric Cryptography	• End-to-end authentication • Hash chains	• Resource consumption • Denial of Service • Route table attack • Replay
ARAN [Sanzgiri et al, 2002]	Asymmetric Cryptography	• End-to-end authentication • Certificate Authority	• Route table attack • Replay attacks
SAODV [Zapata, 2002]	Hybrid Cryptography	• End-to-end authentication • Hash chains • Digital Signatures	• Denial of Service • Route table attack • Replay
SLSP [Papadimitratos & Hass, 2003]	Hybrid Cryptography	• Secure neighbour discovery • Authenticated link state updates	• Denial of Service • Route table attack • Replay
ODSRP [Awerbuch et al, 2008]	Reputation Based	• Path specific reputation lists • Digital Signatures	• Denial of Service • Route table attack • Replay • Byzantine Failures
CONFIDANT [Buchegger & Boudec, 2002]	Reputation Based	• Node specific reputation lists	• Black Hole • Replay

Table 2. Summary of security analysis for secure routing in ad hoc networks

Asymmetric Cryptography

The only solely asymmetric cryptographic approach investigated is the Authenticated Routing for Ad hoc Networks protocol (ARAN). Asymmetric cryptographic is computationally costly compared to symmetric cryptography and it requires the existence of a trusted third party or self organized key management system.
ARAN provides end-to-end authentication for an on-demand mobile ad hoc network. ARAN provides authentication and protecting from replay attacks and unauthorized routing table attacks. ARAN is vulnerable to flooding attacks. A malicious node can flood nodes with fake routing packets signed with illegitimate keys this will result in many unsuccessful verifications and ultimately denial of service and resource consumption.

Hybrid Cryptography

The SAODV and SLSP protocols are hybrid solutions which employ both asymmetric cryptography and symmetric cryptography. The common approach is for all the mutual fields to be digitally signed and the immutable fields, like the hop count, to be protected using hash chains. The Secure Ad Hoc On-demand Distance Vector protocol (SAODV) employs this tactic to provide end-to-end authentication but at the cost of extending the routing packet header. SAODV protects against impersonation attacks on the routing protocol. It also helps prevent replay and denial of service attacks.

Secure Link State Routing Protocol (SLSP) provides secure neighbour discovery and authenticated link state updates but lack a secure data transmission protocol. The Neighbour Location Protocol of SLSP protects against impersonation type attacks where malicious nodes adopting conflicting IP and MAC addresses would want to corrupt the routing table. Furthermore flooding attacks, which result in resource consumption and a denial of services, are prevented by a priority ranking scheme.

Reputation Based

Reputation based or conduct based systems allow for a nodes behaviour in the network to affect its assigned security or trustworthiness. Reputation based protocol can be computationally costly because they usually require packet monitoring systems and the proactive exchange or behavioural evidence between nodes. The On-demand Secure Routing Protocol Resilient to Byzantine Failures (ODSBR) and the CONFIDANT protocol are reputation based systems.

ODSBR uses reputation based system to select the most secure routes. A fault detection method monitors the success of each packet transmission and faults are logged against specific paths. Reputation is path specific in ODSBR. ODSBR couples with asymmetric cryptographic approach to provide end-to-end authentication along the selected secure path. The CONFIDANT uses exclusively reputation based techniques to provide security. Similarly to ODSBR only negative evidence is considered. Nodes monitor every packet which they forward and maintain a local rating list. Reputation or ratings are node specific unlike ODSBR. Both CONFIDANT and ODSBR monitor packet forwarding this will protect the system from black hole attacks. Replay attacks which use the method flooding are prevented using path reputation and node reputation in ODSBR and CONFIDANT respectively. A disadvantage of the negative reputation approach for ODSBR and CONFIDANT is that black list nodes or faulty path entries have an expiration time after which their confidence is reinstated. This allows malicious nodes to continue disrupting the network until they are caught again.

b. Operational Requirements

The presented secure ad hoc routing protocols have certain assumptions that each makes to realize its design. Furthermore protocols are designed specifically for operation in specific routing environments. This section summarizes the operational requirements of the presented secure ad hoc routing protocols. Table -3 summarizes this discussion.

The symmetric cryptographic approaches do not rely on a public key infrastructure but still require some kind of key management in the ad hoc network. The SEAD protocol is designed for a table-driven routing protocol and is based on the DSVD routing protocol. SEAD requires a key management mechanism to distribute an authenticated initial hash element. Ariadne is a DSR based on-demand protocol which assumes there are shared secrets between each communication pair. The shared keys are used in TESLA authentication and a key management system is assumed present to distribute the keys. TESLA authentication also requires time synchronization between each node. This is difficult without the presence of an online TTP.

ARAN, SAODV, SLSP and ODSBR use asymmetric cryptography and key management is simply assumed for each of these protocols. ARAN assumes that an online TTP is present that acts as a certificate authority (CA). Prior shared secrets are assumed between all participating nodes and the CA. ARAN is an on-demand protocol. SAODV protocol is based