

Chapter XIII

Security in Ad-Hoc Networks

Muhammad Mahmudul Islam
Monash University, Australia

Ronald Pose
Monash University, Australia

Carlo Kopp
Monash University, Australia

ABSTRACT

Due to the nature of wireless media, dynamic network topology, resource constraints, and lack of any base station or access point, security in ad-hoc networks is more challenging than with cabled networks. In this chapter, we discuss various attacks on the network layer of ad-hoc networks. We also review security protocols that protect network layer operations from various attacks.

INTRODUCTION

An ad-hoc network consists of a set of nodes that communicate using a wireless medium over single or multiple hops and do not need any pre-existing infrastructure such as access points or base stations. Ad-hoc networks can comprise of mobile, static, or both types of nodes. Ad-hoc networks containing mobile nodes are known as MANETs (mobile ad-hoc networks). An example of ad-hoc networks with static nodes is SAHN

(suburban ad-Hoc network) (Kopp & Pose, 1998). Since ad-hoc networks can be rapidly deployed, they are attractive for digital communication in battlefields, rescue operations after a disaster, and so forth. Ad-hoc networks are also useful in civilian forums for running demanding multimedia applications such as video conferencing.

Due to the lack of a clear physical boundary, a node in an ad-hoc network is very likely to hear the transmissions of a neighbouring node operating in the same frequency channel. If the

node cannot distinguish the packets transmitted by an authorised neighbour from the ones transmitted by a malicious node, then the malicious node can: (1) cause the node to accept misleading information, and (2) propagate unnecessary traffic and misleading information to other parts of the network. As a result normal network operation could be disrupted.

The wireless medium makes eavesdropping easier than with a cabled network. If the packets are not encrypted properly, eavesdroppers can make unauthorised use of the received information and cause trouble. For example, an eavesdropper can forward unencrypted routing information to an accomplice to disrupt the normal operation of the network.

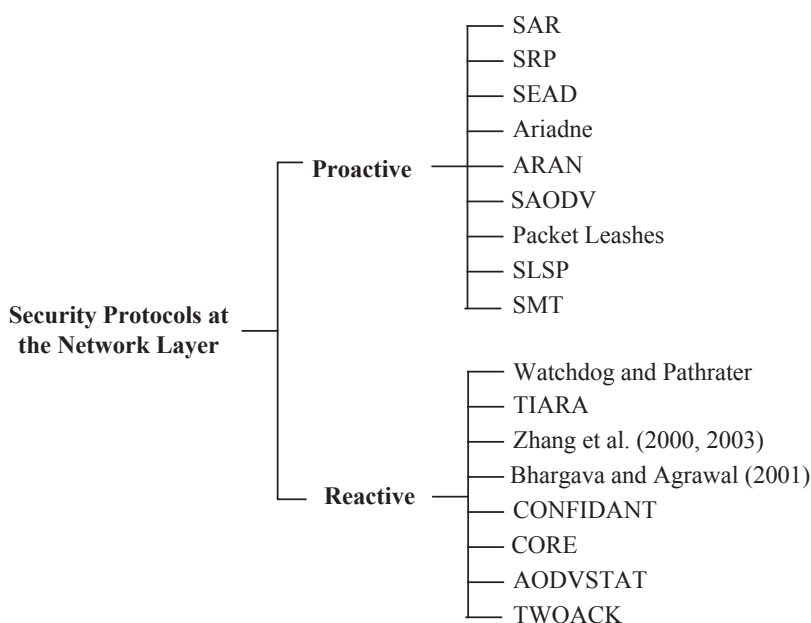
For the aforementioned reasons security is a primary concern in ad-hoc networks in order to provide secure communication among the nodes in a potentially hostile environment (Yang, Luo,

Ye, Lu, & Zhang, 2004). Resource constraints (e.g., battery or computational power), dynamic network topology, and lack of infrastructure (e.g., fixed trusted nodes, base stations, or access points) make the security issue more challenging.

Existing ad-hoc routing protocols, such as DSR (dynamic source routing) (Johnson & Maltz, 1996) or AODV (ad-hoc on-demand distance vector) (Perkins & Royer, 1999) assume a trusted and cooperative environment. These routing protocols have to be protected from malicious nodes that can disrupt the network operation by intentionally disobeying of the protocol specifications (Yang et al., 2004).

There are two main approaches to secure a network: (1) pro-active and (2) reactive. A pro-active approach tries to prevent any attacks happening in the first place. On the other hand, a security protocol using a reactive approach detects any anomaly in network operation and

Figure 1. Classification of network layer security protocols



reacts accordingly. The latter group of protocols is also referred to as the intrusion detection system (IDS). See Figure 1 for the classification and list of network layer security protocols that we review in this chapter.

The rest of the chapter is organised as follows. In the next section, we discuss possible malicious attacks in the network layer of ad-hoc networks. Then we give an overview of the various authentication primitives used in the security protocols and review various key management protocols required for distributing and updating keys in ad-hoc networks. Next we present various pro-active approaches based secure routing protocols. Following this section, we review other security protocols that use the reactive approach to secure ad-hoc networks. We also outline open research challenges in the security domain of ad-hoc networks.

POSSIBLE ATTACKS

This section gives an overview of the attacks pertaining to the network layer.

Drop Packets

A malicious node may disrupt the normal operation of a network by dropping packets (Gupte & Singhal, 2003). This type of attack can be classified into two groups:

- **Black-Hole Attack:** In this type of attack, an attacker drops all types of packets.
- **Grey-Hole Attacks:** Here an attacker selectively drops packets (e.g., only data packets).

Delay Transmissions

A malicious node can give preference in transmitting its own or friends' packets by delaying packets in its output queue. If the packets belong

to any time sensitive application, such as real-time video, they may become useless if not received at the receiver in due time.

Protocol Field Modification

A router is supposed to modify various protocol fields in the messages passing through itself according to the associated protocol specification. Existing resource allocation, transmission control, or routing protocols are based on these assumptions. Here are some of the attacks that can be launched by a malicious node exploiting these assumptions to cause disruption in the network.

- **Divert Traffic through Malicious Node:** The path length to a destination in protocols like AODV (Perkins & Royer, 1999) is measured by the hop count field of the route reply packets. A malicious node can advertise a shorter hop count to a destination by decreasing the actual hop count value in the route reply packet. Thus it succeeds in diverting all the traffic for that destination through itself (Sanzgiri, Dahill, Levine, Shields, & Belding-Royer, 2002). Since the malicious node has become part of the routing path, it can do anything with the packets diverted to it, such as dropping or delaying packets.
- **Divert Traffic through Endless Loop:** Protocols like DSR (Johnson & Maltz, 1996) require the inclusion of the list of intermediate nodes in each packet, known as the source route, for routing packets. A malicious node can become part of the routing process and modify the source route to create loops (Sanzgiri et al., 2002). Consequently it can succeed in launching a DoS attack against the destination of the route since the destination will not receive the packets.
- **Divert Traffic through Bottleneck Node:** A QoS protocol may send probe packets

along a set of possible paths for a particular destination to gather information about how much bandwidth is available along each path. The path having the most bandwidth available may be selected as a route for the new flow. A QoS protocol usually assumes that the nodes, which are on the routing path, set the fields in the probe packets. Moreover, a node is not supposed to modify a value that has been set by another node.

If a malicious node becomes an intermediate node of any of the paths that do not have enough bandwidth available for the new flow, it can increase the available bandwidth values set by other nodes in such a way that the path will ultimately be selected as the routing path for the new flow. When traffic starts to flow along the selected path, it is very likely that all flows passing through the bottleneck node/nodes will be impeded and hence the overall network performance will degrade.

Message Fabrication

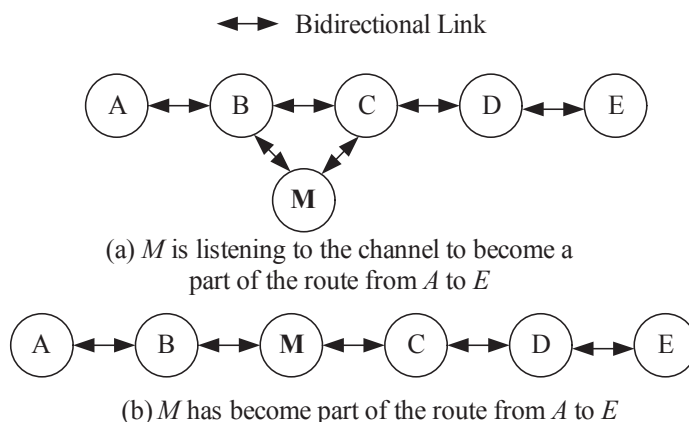
Sanzgiri et al. (2002) have defined message fabrication as the generation of false routing messages.

However, we have redefined this term to expand its scope. That is, in our work message fabrication would imply the generation of any (except for the MAC (medium access control) control messages) false control message. Message fabrication can take any of the following forms:

- **Divert Traffic through Malicious Node:** We have already discussed this type of attack previously that occurs when a malicious node becomes part of the route reply paths and makes changes in the routing protocol fields. Here we present other ways that a malicious node may divert traffic through itself. We will explain these mechanisms in terms of DSR.

Nodes in DSR learn new routes by overhearing transmissions from neighbouring nodes. A malicious node can use this vulnerability to announce false routes that are shorter than existing routes to redirect traffic through itself. For example (see Figure 2), a malicious node *M* can transmit spoofed packets with a shorter source route to *E* via itself. *B* may add this route to its route cache and divert all traffic, destined for *E*, to *M*. Thus *M* becomes part of the routing path and can

Figure 2. The malicious node *M* is becoming a part of the routing path from *A* to *E*



drop or delay subsequent data packets. This particular attack is also known as “route cache poisoning” (Sanzgiri et al., 2002).

- **Force Termination of Ongoing Flows:** Some routing protocols, such as AODV and DSR, facilitate route maintenance by sending error messages from the nodes preceding broken links. Since these messages are not authenticated in traditional AODV or DSR, a malicious node can send a false error message against a working node W so that the source node, sending packets to a destination through W , deletes its corresponding route entry (Sanzgiri et al., 2002). If the source node does not have any other route to the destination, the malicious node can succeed in launching a DoS attack by stopping the ongoing traffic towards that destination.
- **Routing Table Overflow:** A node can contain a finite number of route entries in its route table. In reactive routing protocols, a malicious node can try to overflow route tables of other nodes by initiating route discovery for non-existent nodes (Sanzgiri et al., 2002). In pro-active routing, the same purpose can be achieved by broadcasting route information of non-existent nodes (Sanzgiri et al., 2002). In this way, the malicious node can prevent other nodes adding legitimate route entries in their route tables.
- **Denouncing a Benign Node:** A malicious node can report to other nodes of the network that a benign node is misbehaving. If the report is not authenticated, the benign node may not get any legitimate service from other members of the network.

Replay Attack

In this attack a malicious node intercepts a message and retransmits it at a later time, modified or unmodified. By replaying stale information the malicious node can cause inconsistency in the network and consequently can prevent the network

from operating properly. A repetitive replay attack can blind a target node with a storm of traffic so that other nodes are prevented from getting any service from the attacked node.

Broadcast Storm Attack

A malicious node can send network-wide broadcast packets to a node within its transmission range. If the packets are not authenticated, the node will rebroadcast the packet to its neighbourhood. If this process continues, the whole network can be flooded with so much traffic that transmission contention (if the network uses a contention-based MAC protocol) and network congestion may become inevitable. Consequently legitimate flows may not be able to get their required QoS.

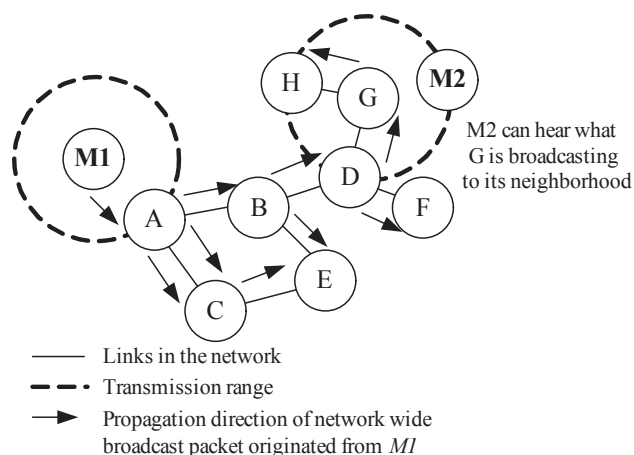
Piggyback Attack

If a malicious node knows how packets are routed in a network, it can piggyback its messages on network-wide broadcast packets to send to its accomplice located on another part of the network. If the malicious node sends a large number of packets within a short-time period, the network may face problems similar to the broadcast storm attack. Figure 3 illustrates a piggyback attack.

Tunnelling Attack

This is also known as the *wormhole attack* (Hu, Perrig, & Johnson, 2003) or *route hijacking* (Ramanujan, Ahamed, Bonney, Hagelstrom, & Thurber, 2000). In this attack one or more malicious nodes link two parts of a network through a path that may seem shorter in distance or duration than would otherwise be expected (Sanzgiri et al., 2002). This attack does not require the malicious nodes to have any knowledge of cryptographic keys (Gupte & Singhal, 2003). The results of this attack could be diverting traffic through malicious nodes or preventing nodes from discovering paths more than a certain length.

Figure 3. Piggyback attack where a malicious node M1 is using the network as a carrier for sending messages to another malicious node M2



Identity Theft

In the absence of any authentication mechanism a malicious node can easily masquerade as a legitimate node by changing its identity (e.g., IP and MAC addresses) to that of the legitimate node (Sanzgiri et al., 2002). In this way the malicious node can perform any of the attacks mentioned so far without being detected.

AUTHENTICATION PRIMITIVES

Existing network layer security protocols for ad-hoc networks depend on one or more of the following authentication primitives (Yang et al., 2004):

- Hashed message authentication code (HMAC),
- Digital signature, and
- One-way hash chain.

Hashed Message Authentication Code (HMAC)

HMAC requires the communicating nodes to negotiate a symmetric key prior to sending any packet. This key and some parts of the transmitted packet are fed into a one-way hash function to generate a message authentication code. This code is then attached to the transmitted packet. The intended receiver recomputes a message authentication code of the received packet with the same key and hash function. If the recomputed code and the attached code match, the receiver can be assured that the packet originated from an authentic node. The computation involved in this process is very low due to the use of a hash function and symmetric cryptosystems.

Digital Signature

A digital signature may be based on asymmetric cryptography such as RSA (Rivest Shamir Adle-

man) (Rivest, Shamir, & Adleman, 1978). In this scheme the sender of a packet signs the packet with its private key and the receivers verify the authenticity of the packet using the associated public key. To make the receivers believe that the public key belongs to an authentic sender, the public key needs to be signed by trusted entities. Digital signatures involve more computational overhead for authentication and encryption than a symmetric crypto-system (Yang et al., 2004).

One-Way Hash Chain

In a one-way hash chain a one-way hash function H maps an input of arbitrary length to a fixed length bit string. The function of H is expected to be simple yet robust enough not to be invertible by finite computational capability. To create a hash chain of length $n + 1$, a node chooses a random x where x is a variable length bit string. H is applied to x repetitively to compute a chain $h_0, h_1 \dots h_n$ where $h_0 = x$ and $h_i = H(h_{i-1})$ for $0 \leq i \leq n$. For a given element in a hash chain, it is possible to verify the elements having lower subscript values of that chain. For example, given h_i , a node can authenticate h_{i-3} by computing the value of $H(H(H(h_i)))$ and verifying that the computed value equals h_i . The computational requirement for hash chain-based authentication is lower than for digital signatures. Coppersmith and Jakobsson (2002) describe efficient mechanisms for storing and generating hash chains.

KEY MANAGEMENT IN AD-HOC NETWORKS

Keys are needed for authentication and encryption of transmitted messages. The primary goals of a key management system are to distribute the keys to nodes securely, and to update them periodically or on-demand. Due to the lack of infrastructure, key management in ad-hoc networks is a challenging task. Below some well-known

key management protocols suitable for ad-hoc networks have been described.

Threshold Cryptography

Public key infrastructure can be used for easy distribution and certification of keys. A certification authority (CA) certifies the public key of a node by signing the public key and identity of the node with the CA's private key. The CA has to be online all times, which may not be feasible for ad-hoc networks. A CA consisting of a single node is vulnerable to a single point of failure and compromise. If the CA is replicated to multiple nodes, it will not be vulnerable to a single point of failure any more. However it would still be susceptible to a single point of compromise.

Zhou and Haas (1999) have proposed to distribute the CA functionality to several nodes using threshold cryptography. In a (n, m) threshold cryptography scheme, the public key of the CA is known to all nodes, the private key of the CA is shared among n nodes called servers according to the secret sharing model of Shamir (1979), and a minimum of m shares of the private key are needed to create a certificate. Hence this scheme can withstand up to $n-m$ compromised nodes.

Yi and Kravets (2002) have extended the work of the (n, m) threshold cryptography scheme. They have discussed which nodes to select as servers and how to make the servers available to other nodes efficiently without requiring too much network overhead. They have suggested that physically more secure and computationally more powerful nodes should be selected as servers. If a node has enough cached routes towards the servers, it should send requests to the servers for certificates without employing any broadcast-based route discovery mechanism. Using unicast traffic may reduce network overhead that would otherwise increase if a broadcast-based route discovery mechanism were used.

Luo and Lu (2000) also use a threshold cryptography scheme but allow any m nodes to

carry a share of the private key of the CA. Hence this scheme provides a fairer distribution of the burden of creating certificates than the schemes discussed earlier. Since any m nodes in the local neighbourhood of a requesting node can create a certificate, communication overhead is minimised. This scheme allows any node not possessing a share of the private key of the CA to obtain a share from any group of m nodes that jointly possess the private key. This feature can make the scheme vulnerable to Sybil attack (Douceur, 2002) where a malicious node can take as many identities as necessary to collect enough shares and reconstruct the CA's private key (Capkun, Buttyan, & Hubaux, 2003).

Self-Organised Public Key Management for MANETs

Capkun et al. (2003) have proposed a self-organised public key system for ad-hoc networks that does not need any CA or special nodes. Each node issues certificates to others based on personal acquaintances.

Each node stores the certificates it has issued to others and others have given it. Periodically neighbours exchange new certificates. Two certificates are said to be conflicting if they contain the same public key or if they belong to the same node. A conflict may arise because a malicious node has issued a false certificate. If a node finds any of the certificates is conflicting, it labels the certificate as conflicting. To resolve the dispute, the node obtains certificates from other nodes and takes a majority decision on the collected information.

When a node X wants to obtain the public key of another node Y , X acquires a chain of valid public key certificates such that: (1) the first certificate can be directly verified by X using a public key that X holds and trusts; (2) each remaining certificates can be verified using the public key contained in the previous certificate of the chain; and (3) the last certificate contains the public key of Y .

Since the key authentication is performed via chains of public key certificates, the protocol assumes the trust is transitive, that is, if A trusts B and B trusts C , then A trusts C .

SECURE ROUTING IN AD-HOC NETWORKS

Secure routing protocols try to prevent malicious nodes from taking part in any routing process (e.g., sending updated information) and becoming a part of the routing path. Next we review some of the well-known secure routing protocols.

Secure Aware Ad-Hoc Routing (SAR)

SAR (Yi, Naldurg, & Kravets, 2001) outlines various alternatives to authenticate and encrypt routing packets. It incorporates a trust hierarchy in the network in order to associate different privileges or "trust levels" with different nodes.

In SAR, the headers of route request and reply packets are signed and encrypted. Since only nodes with the same level of trust share the same key, any node that does not belong to the same trust level cannot decrypt the packets and hence is forced to drop the packets. The discovered route may not be the shortest for a particular source and destination pair but may be the most secure in terms of the desired trust level.

An implementation of SAR in AODV has been provided. The headers of route request and reply packets are signed using digital signatures and encrypted using the key shared among all the nodes having the same level of trust.

SAR relies on certification authorities for distributing keys in the network. If a centralised certification authority is used, it has to be online all the time and could be a single point of failure that would make the whole system ineffective. If a distributed certification authority is employed then more than one trusted node would be required to construct a certificate. Depending on

the distance in the number of hops, the neighbourhood density and the number of the trusted nodes constituting the certification authority, and the update frequency of the keys, the associated communication overhead and response time of getting a reply from the certification authority may not be negligible.

Secure Routing Protocol (SRP)

SRP (Papadimitratos & Haas, 2002) ensures that each pair of end nodes can differentiate between legitimate requests and replies. This is realised through a security association between the two end nodes. A security association can be achieved by establishing a shared key (K_{ST}) between the end nodes. Both the requests and replies are broadcast to thwart potentially malicious nodes from controlling them to reach their destinations.

The SRP adds a header, called the SRP header, to the underlying routing protocol. Three fields (Q_{ID} , Q_{SEQ} , and SRP_{HMAC}) in the SRP header are used to authenticate the requests and replies at the end nodes. The Q_{ID} is a 32-bit query identifier generated by the source and used by the intermediate nodes to identify duplicate requests. The Q_{SEQ} is a monotonically increasing 32-bit sequence number. It is negotiated during the security association and maintained by the source node for each destination. The SRP_{HMAC} is a 96-bit HMAC calculated by the source using K_{ST} over the addresses of the source and destination nodes, Q_{ID} , and Q_{SEQ} .

When the destination receives a request, it checks Q_{SEQ} for outdated or replayed requests. It recomputes the HMAC using K_{ST} over the message it receives. If the recomputed value and the received SRP_{HMAC} are the same, the destination can be assured the route request originated from the source and has not been tampered with. Otherwise the destination drops the request.

If the destination accepts the packet as a valid route request, it puts the accumulated route into a reply packet. It puts the received values of Q_{ID} and Q_{SEQ} into the reply packet. It calculates a

message authentication code using K_{ST} in the same way the source did and puts this value in the reply packet. This enables the source of the route request to verify the integrity and authenticity of the reply packet.

Each node monitors the rate at which each of its neighbours forwards route request packets. It gives priority to the neighbours that less frequently forward request packets. This mechanism attempts to limit the flooding of route request packets from malicious nodes.

SRP does not have intermediate nodes checking the authenticity of route requests and route replies. This means that a malicious node can become part of a routing path and later disrupt the operation of the routing protocol. To handle the problem of malicious nodes flooding the network with unnecessary route requests and replies, SRP employs a system at each node that monitors the rate at which each of the neighbours forwards route request packets. Each node forwards packets, with lower priorities for the neighbours that are found sending excessive route requests or route replies. This monitoring system may exacerbate greedy behavior of selfish nodes. A selfish node may misbehave by not forwarding route request packets originated from other nodes. Its neighbours will consider it a low frequency route request packet forwarder and give it priority. Consequently this selfish node may achieve better performance than benign nodes. Moreover SRP does not protect route error messages. Therefore a malicious node can send false route error messages to isolate a node from the network.

Secure Efficient Ad-Hoc Distance (SEAD) Vector Routing Protocol

SEAD (Hu, Johnson, & Perrig, 2002a) aims to protect the route updates of DSDV (destination sequenced distance vector) (Perkins & Bhagwat, 1994) from malicious attacks. DSDV is a table driven or pro-active routing protocol.

A node's shortest distance, usually expressed in hops, is referred to as the metric in DSDV. A node accepts a route update packet containing updated route information for a particular destination if the sequence number included in the packet is greater than the corresponding entry currently available in the node's routing table. The node also accepts the packet if the sequence number is equal but the new metric is lower than those entries in its routing table. A malicious node can send route updates with higher sequence numbers, or with the same sequence number and lower metric for the destination. Neighbours of the malicious node receiving these updates may believe that the malicious node has the shortest or the updated path to that destination. This will make the malicious node the first hop towards that destination from those neighbours. Consequently traffic destined for that destination will be forwarded to that malicious node. The malicious node can then disrupt the flow of the traffic by dropping or delaying packets. SEAD uses a one-way hash chain to prevent malicious nodes from providing incorrect route updates containing a higher sequence number or the same sequence number with lower metrics.

In SEAD each node generates its one-way hash chain of length n and divides the hash chain into groups of m elements each. The value of m is chosen in such a way that the maximum diameter of the network is $m-1$. Each group corresponds to a sequence number, that is, for a new sequence number the previously used elements are replaced by a new group of elements. Before a node can use a group of elements for authentication, it has to distribute the element with the highest subscript value of that group securely to all other nodes in the network.

Here is an example that illustrates how SEAD works. A destination node X wants to send its updated route information to its neighbours in a route update packet. X sets the value of the metric in the route update packet to 0. The sequence number field is set to a new value, say i . Assume

the generated hash chain at X consists of the sequence h_0, h_1, \dots, h_n and $k = n/m - i$. SEAD requires X to send $h_{mk+(m-1)}$ to all other nodes in the network before using this value to authenticate $h_{mk}, h_{mk+1}, \dots, h_{mk+(m-2)}$. X puts the first value h_{mk} from the selected group in the route update packet.

A node receiving a route update has to apply the hash function on the received hash value $(m-1-(value\ of\ the\ metric))$ times. If the recomputed hash value matches the previously sent $h_{mk+(m-1)}$, the receiving node can assume that a malicious node has not increased the sequence number or lowered the value of the metric keeping the sequence number unchanged. For example when a neighbour A receives the packet it applies the hash function $(m-1-0)$ times, where 0 is the value of the metric in the packet, on the received h_{mk} to get $h'_{mk+(m-1)}$. If $h'_{mk+(m-1)} = h_{mk+(m-1)}$, A accepts the sequence number and the metric value of the route update packet as valid. Otherwise A ignores the packet.

When A rebroadcasts the route update packet, it increases the metric value by 1, keeps the sequence number field as it is and replaces the hash value with $h_{mk+1} = H(h_{mk})$.

When a neighbouring node B receives this packet, it applies the hash function on the received h_{mk+1} , $m-1-1$ times to get $h''_{mk+(m-1)}$. If $h''_{mk+(m-1)}$ matches $h_{mk+(m-1)}$, B accepts the information contained in the route update packet as valid and considers the packet for further processing. Before rebroadcasting, B increases the received metric value by 1, keeps the sequence number field unchanged and replaces h_{mk+1} with $h_{mk+2} = H(h_{mk+1})$. This process of validating and rebroadcasting is repeated at each node that receives the route update packet.

If a node decreases the metric value of a route update packet, it also has to replace the hash value of the packet with an earlier hash value. Due to the one-way nature of the hash function, this is not possible. On the other hand, if a node increases the sequence number in a route update packet, it must replace the hash element of the packet with a hash element that belongs to the next group of

the corresponding hash chain. Though the node can apply H on the hash element of the packet an appropriate number of times to get the new element suitable for a higher sequence number, it will still fail to convince others to accept the packet. This is because the node, which initially sent the route update packet, has not yet distributed the authentic hash element from the next group to other nodes.

SEAD requires authenticated elements from the hash chain of each node to be distributed to all other nodes prior to sending update packets. It cannot prevent the “same distance attack” where a malicious node rebroadcasts a route update with the same sequence number and hop count, but with a different sender address (e.g., replacing the original sender address with its own). A secured layer that can identify any unauthorised packet modification is needed to identify such attacks.

Ariadne

Nodes using Ariadne (Hu, Perrig, & Johnson, 2002b) can authenticate routing messages using any of the three schemes: (a) shared secrets between each pair of nodes, (b) shared secrets between communicating nodes combined with broadcast authentication, or (c) digital signature. Here we discuss Ariadne based on shared secrets between communicating nodes combined with a broadcast authentication protocol called the timed efficient stream loss-tolerant authentication (TESLA) (Perrig, Canetti, Tygar, & Song, 2000).

A broadcast authentication protocol is used to authenticate broadcast packets sent to multiple nodes. In a traditional broadcast authentication protocol the sender distributes a public key to all the intended receivers from its generated public and private key pair. Then it includes a message authentication code with each packet. The message authentication code is computed using the private key. The receivers can verify the authenticity of the received packets by recomputing the

message authentication code using the public key and comparing this value with the one included in the packet. If they match, the receiver assumes that the packet has been sent from the sender. TESLA differs from this traditional asymmetric protocol in that it achieves this asymmetry using clock synchronisation and delayed key disclosure, rather than from computationally expensive encryption/decryption functions.

To use TESLA for authentication each sender generates a one-way hash chain and pre-determines a schedule at which it discloses each key of its hash chain to other nodes. The sender discloses keys in reverse order, that is, in order of h_n, h_{n-1}, \dots, h_0 . A simple key disclosure schedule, for example, would be to disclose key h_i at time $T_0 + i \times t$ where T_0 is the time at which h_n has been disclosed and t is the key disclosure interval.

TESLA assumes each receiver can determine which keys a sender might have already disclosed based on loose time synchronisation between nodes. Let Δ be the maximum time synchronisation error between any two nodes, which is known to all nodes. When a sender sends a packet, it adds an HMAC of the packet using a key h_j , which has not been used before.

When a receiver receives the packet it verifies that h_i cannot yet have been disclosed. For example, if the packet arrival time at the receiver is t_r and the receiver knows that the earliest time at which the sender will disclose h_i is $T_0 + i \times t$, the receiver needs to verify that $t_r \leq (T_0 + i \times t - \Delta)$ implying h_i has not yet been disclosed. If the check is not successful, the receiver discards the packet since the sender may already have disclosed h_i and a malicious node may have forged the packet. Otherwise the receiver buffers the packet and waits for the sender to disclose h_i . When the receiver receives h_i , it first authenticates h_i using the mechanism outlined in the *One-way Hash Chain* section. Then it authenticates buffered packets authenticated with h_j , where $j \geq i$.

Ariadne’s basic idea of route discovery and maintenance is based on DSR. For end-to-end

authentication Ariadne uses HMAC using shared keys. Let S denote the source that performs a route discovery for destination D , and K_{SD} and K_{DS} denote the keys they share for message authentication in each direction respectively. Setting up shared keys between the source and all intermediate nodes on a route discovery path may be expensive for some ad-hoc networks. Hence it has been proposed to use TESLA for authenticating the intermediate nodes.

A route request (RREQ) packet in Ariadne contains eight fields: *REQUEST*, *initiator*, *target*, *id*, *time interval*, *hash chain*, *node list*, and *HMAC list*. When S sends a RREQ, it sets the *initiator* and *target* to the addresses of S and D respectively. S sets the *id* to an identifier that can be used to distinguish between duplicate RREQs. S sets the *time interval* to the pessimistic expected arrival time t_i of the RREQ at D . S initialises the hash chain to h_0 where $h_0 = \text{HMAC}[\text{REQUEST}, S, D, id, t_i]K_{SD}$. Here $\text{HMAC}[\dots]K$ denotes the HMAC of $[\dots]$ computed using K . S does not initialise the *node list* and the *HMAC list*. The RREQ appears as follows:

$$S \rightarrow *: \langle \text{REQUEST}, S, D, id, t_i, h_0, (), () \rangle$$

When a neighbour A of S receives the RREQ, it checks the $\langle S, id \rangle$ to determine if it has already seen the same RREQ. If it has, it discards the packet. It also verifies t_i by checking that t_i is not too far in the future and that the corresponding hash chain value has not yet been disclosed. If the *time interval* value is not valid, A discards the RREQ. Otherwise A modifies the RREQ as follows. A appends its address to the *node list* field. It updates the hash chain field with $h_1 = H[A, h_0]$ and appends the $M_A = \text{HMAC}[\text{REQUEST}, S, D, id, t_i, h_p(A), ()]h^A_i$ to the *HMAC list*. Here H is a one-way hash function known to all nodes and h^A_i is the TESLA key of the hash chain generated by A . Then A rebroadcasts the modified RREQ.

$$A \rightarrow *: \langle \text{REQUEST}, S, D, id, t_i, h_1, (A), (M_A) \rangle$$

When A 's neighbour B receives the RREQ from A , it verifies the RREQ as discussed earlier. If the RREQ is valid, B calculates $h_2 = H[B, h_1]$ and $M_B = \text{HMAC}[\text{REQUEST}, S, D, id, t_i, h_2, (A, B), (M_A)]h^B_i$. Then it rebroadcasts a modified RREQ that looks as follows:

$$B \rightarrow *: \langle \text{REQUEST}, S, D, id, t_i, h_2, (A, B), (M_B) \rangle$$

Assume that this RREQ goes through node C before reaching destination D . The RREQ broadcast from C appears as follows:

$$C \rightarrow *: \langle \text{REQUEST}, S, D, id, t_i, h_3, (A, B, C), (M_C) \rangle$$

Here $h_3 = H[C, h_2]$ and $M_C = \text{HMAC}[\text{REQUEST}, S, D, id, t_i, h_3, (A, B, C), (M_A, M_B)]h^C_i$.

When D receives the RREQ, it checks the validity of packet determining that the keys from the *time interval* specified have not yet been disclosed and the hash chain field is equal to $H(C, (H(B, (H(A, \text{HMAC}[\text{REQUEST}, S, D, id, t_i]K_{SD}))))))$.

If the RREQ is valid, D returns a route reply (RREP) packet to S containing eight fields: *REPLY*, *target*, *initiator*, *time interval*, *node list*, *HMAC list*, *target HMAC*, *key list*. The values of *target*, *initiator*, *time interval*, *node list*, and *HMAC list* are set to the values from the RREQ packet. The *target HMAC* is set to $M_D = \text{HMAC}[\text{REPLY}, D, S, t_i, (A, B, C), (M_A, M_B, M_C)]K_{DS}$. The key list is kept empty at D .

A node forwarding a RREP waits until it is able to disclose its key from the specified time interval. Then it appends its key to the *key list* field.

The RREPs forwarded by D , C , B , and A contain the values as shown in the following:

$$D \rightarrow C: \langle \text{REPLY}, D, S, t_i, (A, B, C), (M_A, M_B, M_C), M_D, () \rangle$$

$$C \rightarrow B: \langle \text{REPLY}, D, S, t_i, (A, B, C), (M_A, M_B, M_C), M_D, (h^C_i) \rangle$$

$$B \rightarrow A: \langle \text{REPLY}, D, S, t_p, (A, B, C), (M_A, M_B, M_C), M_D, (h_p^C, h_p^B) \rangle$$

$$A \rightarrow S: \langle \text{REPLY}, D, S, t_p, (A, B, C), (M_A, M_B, M_C), M_D, (h_p^C, h_p^B, h_p^A) \rangle$$

When S receives the RREP, it verifies that each key in the *key list*, M_D and each HMAC in the *HMAC list* are valid. If tests are successful, S accepts the RREP. Otherwise S discards the packet.

Ariadne also authenticates route error packets using the TESLA protocol.

TESLA and Ariadne require a sender node to generate a one-way hash chain and attach a hash element from the hash chain to each transmitted packet. Then at a later time the sender discloses an element from its hash chain that can be used by the intended receivers to authenticate the previously attached element. The delayed disclosure mechanism requires clock synchronisation. Receivers may have to buffer packets before the packets can be verified with the delayed release of the authentic hash value. The buffering of packets can increase the response time of the routing protocol (Yang et al., 2004). Moreover the release of the authentic hash values involves a second round of communication, increasing network traffic.

Authenticated Routing for Ad-Hoc Network (ARAN)

ARAN (Sanzgiri et al., 2002) is a secure on-demand routing protocol based on AODV (Perkins & Royer, 1999). It uses certificates from trusted entities and public key cryptography for authenticating route request, reply and error packets.

In ARAN each node gets a certificate ($CERT$) from a trusted certificate server T before joining the network. For a node A , the certificate looks as follows: $CERT_A = [IP_A, PBK_A, t, e]PVK_T$. Here IP_A is the IP address of A , PBK_A is the public key of A , t and e are the times indicating when the certificate was created and will expire respectively, and PVK_T

is the private key of T . The term $[...]PVK_T$ denotes that all the values inside the square bracket are concatenated and signed with PVK_T . All nodes are required to contain fresh certificates and must have the public key of T .

To discover routes from A to X , A broadcasts a route request packet with $[RDP, IP_X, CERT_A, N_A, t]PVK_A$. The route request consists of a RDP (route discover packet) marker, the destination's IP address IP_X , certificate of the initiator $CERT_A$, a nonce N_A and the current time t . The nonce is monotonically increased each time a route request is sent. Nodes record the nonce they have seen with the related time-stamp. Therefore they ensure the freshness of the request.

Upon receiving a route request, each intermediate node sets up a reverse path back to the previous node so that it can be used later to unicast the corresponding reply packet back to the initiator.

When an intermediate node B , neighbour of the initiator A , receives the route request with the pair $\langle N_A, IP_A \rangle$ for the first time it extracts A 's public key from $CERT_A$ and validates the signature. If the signature appears valid B rebroadcasts a route request which contains $[[RDP, IP_X, CERT_A, N_A, t]PVK_A]PVK_B$ and $CERT_B$.

When an intermediate node C receives the route request from B for the first time, it verifies the authenticity of the request by validating the signature with the B 's public key. B 's public key is retrieved from $CERT_B$. Then C removes B 's signature and certificate from the route request and rebroadcasts $[[RDP, IP_X, CERT_A, N_A, t]PVK_A]PVK_C, CERT_C$.

Eventually single or multiple route requests with the same $\langle N_A, IP_A \rangle$ pair reach the destination X . The destination replies to the first route request with a given source and nonce pair. The route reply with a REP (Reply) marker from X for A consists of $[REP, IP_A, CERT_X, N_A, t]PVK_X$.

Each intermediate node processes a route reply in the same manner as outlined for route requests except that each reply is unicast rather

than broadcast. The packet formats of route reply packets, while being unicast by C and B , are as follows:

$$C \rightarrow B : [[REP, IP_A, CERT_X, N_A, t]PVK_X]PVK_C, CERT_C$$

$$B \rightarrow A : [[REP, IP_A, CERT_X, N_A, t]PVK_X]PVK_B, CERT_B$$

In ARAN a node generates a signed route error packet if it finds out one of the links of an active route is broken. Here is an example. Assume the route from A to X consists of intermediate nodes B , C , and D . If the link $C \rightarrow D$ gets broken, C unicasts $[ERR, IP_A, IP_X, CERT_C, N_C, t]PVK_B$ to B . B verifies the authenticity of the error packet from the signature and the public key of C . Then it forwards the unmodified error packet to A . The nonce and the time-stamp together ensure the freshness of the message. Since the error packet is signed, A can be assured that it was not sent from a malicious node.

If a certificate $CERT_R$ needs to be revoked, the trusted entity T broadcasts a revocation packet with packet type *revoke*. It appears as $[revoke, CERT_R]PVK_T$. Any node receiving this message rebroadcasts it to its neighbours and avoids using routes passing through the untrusted node.

To reduce the processing overhead using asymmetric cryptography at each node at each time, the authors have suggested neighbouring nodes exchange secret keys using their public keys and certificates, and use the secret keys to sign routing packets (Sanzgiri, Dahill, Levine, Shields, & Belding-Royer, 2005). However the end nodes are still required to include full public key signatures.

If a node, whose certificate is being revoked, is the only means to connect two parts of a network, it may not propagate the certificate revocation message from one part to another and consequently may cause the network to be partitioned (Gupte & Singhal, 2003). Signing each route request or

reply packet by intermediate nodes increases the size of the corresponding packet at each hop. Since ARAN uses a time-stamp together a nonce to ensure the freshness of a message, it may become less effective if clocks of all nodes in the network are not synchronized.

ARAN requires each node to communicate with a certification authority when the node needs to certify its updated public key. Therefore ARAN can suffer from the problems as stated previously for SAR.

Secure AODV (SAODV)

Zapata and Asokan (2002) have proposed SAODV to secure AODV. SAODV uses hash chains similar to SEAD. Unlike SEAD, SAODV does not require each node to disseminate some elements from a chain prior to authenticating earlier values of the hash chain. Unlike ARAN, SAODV uses a hash chain to authenticate the hop count field of route request and route reply packets at each intermediate node. Moreover, it uses a digital signature to sign all fields of route error, and most of the fields (except for the hop count field) of route request and route reply packets at each intermediate node.

To discover routes from S to D , S chooses a maximum hop count MH based on the expected diameter of the network. Then it generates a one-way hash chain h_0, h_1, \dots, h_{MH} , similar to SEAD, of length $MH+1$. Here h_0 corresponds to a random number generated by S and $h_i = H(h_{i-1})$ where H is a hash function known to all nodes in the network. Before broadcasting a RREQ, S includes MH, h_{MH} , the current value of hop count HC ($HC = 0$ for S), and an element h_i from the hash chain that corresponds to $HC = 0$ (i.e. $h_i = h_0$). It also includes a digital signature computed using its private key PVK_S and all fields (except for h_i and HC) of the route request packet. The route request packet after adding the fields is referred to as the RREQ-SSE (route request with single signature extension).

When a node receives a copy of the RREQ-SSE, it calculates $h'_{MH} = H^{MH-HC}(h_i)$. $H^{MH-HC}(h_i)$ means applying the hash function H to the hash value enclosed within the brackets $MH-HC$ times. The node also recomputes the digital signature using S 's public key PBK_s and the fields (except for h_i and HC) of the received RREQ-SSE. If $h'_{MH} = h_{MH}$ and the recomputed digital signature is valid, the node rebroadcasts the RREQ-SSE. Otherwise the node drops the packet. Before rebroadcasting the node updates previous the h_i with $H(h_i)$ and HC with $HC+1$ and performs other tasks required for the normal protocol operation of AODV.

When the RREQ-SSE reaches D , D checks the validity of the RREQ-SSE in a similar manner as described earlier. If the RREQ-SSE is valid, D replies with an RREP-SSE (Route Reply with SSE). The RREP-SSE contains the hash chain values and the signature generated by D .

When a node receives an RREP-SSE, it verifies the integrity and authenticity of the RREP-SSE in a similar manner to an RREQ-SSE. If the RREP-SSE is valid, the node updates the hash and hop count values in the same way as it would do for an RREQ-SSE and follows the normal protocol operation of AODV. Otherwise the node drops the RREP-SSE.

SAODV allows each intermediate node to send a route reply in response to a route request if the node contains a fresh route to the destination. This requires the intermediate node to sign the generated route reply on behalf of the final destination.

To solve this problem SAODV requires each intermediate node to store the last route reply it has forwarded for each destination. If an intermediate node receives a route request destined for D and contains a stored route reply originated from D with a sequence number greater than the sequence number contained in the route request, it can reply with a route reply denoted by RREP-DSE (RREP with double signature extension). The intermediate node puts the protocol fields (including the signature computed by D) of the

stored route reply within the RREP-DSE. The RREP-DSE must have a different value of route lifetime than the one stored in the stored route reply. To account for this change, the intermediate node includes a new lifetime value in the RREP-DSE and signs this new value with its public key. Thus the RREP-DSE contains two signature values, one generated by D and the other one generated by the intermediate node. This is why the term "double signature extension (DSE)" is used instead of the "single signature extension (SSE)". This RREP-DSE is verified and handled by other nodes in the same way as an RREP-SSE would have verified and handled.

SAODV also allows each intermediate node to create a reverse route to the originator S of a route request so that if another node asks for a route to S , the intermediate node can send a reply. This feature also requires the intermediate node to sign the generated route reply on behalf of S .

To address this issue, SAODV requires each source node to include the protocol fields of a route reply and a signature that signs the fields corresponding to the route reply in the route request packet. This extended route request is denoted by RREQ-DSE (RREQ with double signature extension). When a node receives this RREQ-DSE, it considers this packet containing a route request seeking a route to a destination and as well as a route reply sent by the originator of the route request. Therefore, it handles the packet in a similar way to handling RREQ-SSE and RREP-SSE.

SAODV uses digital signatures to identify the originator and forwarders of a route error packet. When a node generates or forwards a route error packet, it signs the whole message with its private key. A node receiving a route error packet verifies previously appended signatures to verify the originator and the forwarders of the route error packet.

SAODV assumes that there is a key management sub-system that enables each node to obtain public keys of other nodes of the same network.

Moreover, each node is assumed to be capable of verifying the association between the identity of a given node and the public key of that node. A general limitation of a hash chain-based approach is that it may be complicated and inefficient for continual metrics that take non-integer values (Yang et al., 2004).

Packet Leashes

Hu et al. (2003) have provided two approaches to handle the tunnelling attack using the notion of a *packet leash*. A leash is some information that is added to a packet to determine if the packet has traversed an unrealistic distance.

There are two types of packet leashes: *geographical* and *temporal*. A geographical leash ensures that the receiver of a packet is within a certain distance from the sender. A temporal leash puts an upper bound on the lifetime of a packet to restrict its maximum travel distance.

With a geographical leash each node knows its location and all nodes have loosely synchronised clocks. Let Δ be the maximum time synchronisation error between any two nodes and v be an upper bound on the velocity of any node. When sending a packet the sender includes its own location p_s and the time t_s at which it sent the packet.

When receiving the packet, the receiver computes an upper bound on the distance d_{sr} between the sender and itself based on Δ , received information, its current location p_r , local receive time t_r , and maximum relative error in location information δ :

$$d_{sr} \leq \|p_s - p_r\| + 2v \cdot (t_r - t_s + \Delta) + \delta$$

In temporal leashes all nodes are assumed to have tightly synchronised clocks such that the maximum difference between any two nodes' clocks is Δ . There are two approaches to construct temporal leashes:

1. When sending a packet the sender includes the time t_s at which it sent the packet. When the receiver receives the packet it records the receive time t_r . Based on the difference between t_r and t_s , and the speed of light the receiver can detect if the packet has traveled too far. If the sender's clock is faster than the receiver's, Δ is added to the difference. Conversely Δ is subtracted from the difference if the sender's clock is slower than the receiver's.
2. When the sender sends a packet it includes an expiration time t_e in the packet after which the receiver should not accept the packet. If t_s is the time at which the sender sent the packet, L is the maximum distance the packet is allowed to travel and c is the speed of light, then $t_e = t_s + L/c - \Delta$. When the receiver receives the packet it has to check if the receive time $t_r < t_e$. If not the receiver drops the packet.

HMAC, digital signature, or other authentication technique such as Schnorr's signature (Schnorr, 1991) or TESLA can be used to authenticate the information pertaining to packet leashes.

The main limitation for "packet leashes" is that the clocks have to be synchronised to handle the tunnelling attack. Moreover packet leashes have not been designed to guard against any other attacks except for the tunnelling attack.

Secure Link-State Protocol (SLSP)

SLSP (Papadimitratos & Haas, 2003a) secures the discovery of neighbours and distribution of link-state information of table driven or pro-active routing protocols using digital signatures and one-way hash chains.

During neighbour discovery in link-state routing protocols, each node broadcasts a HELLO message to its neighbours. The HELLO message

contains a signed copy of the sender's IP and MAC (medium access control) addresses. Receiving nodes validate the signature and store the new neighbour's information. For signature verification SLSP assumes that each node has a public and private key pair where key certification is provided by a certification authority.

Each node contains a neighbour lookup protocol (NLP) which notifies SLSP of the following events: (1) Two MAC addresses are being used by the same IP address; (2) One MAC address is being used by two different IP addresses; and (3) Another node is using the same MAC address as the detecting node. Notifications from NLP enable SLSP to discard the packets having any of these aforementioned criteria.

Link-state updates in SLSP are signed and propagated for a limited number of hops. Similar to SEAD, a hash chain is used to authenticate the hop count value so that an SLSP update does not travel too many hops.

Like SRP, SLSP uses a monitoring system to limit the flooding of link-state updates from malicious nodes. For relaying link-state updates, a node gives priority to those nodes that relay or generate fewer link-state updates than other nodes.

SLSP is only concerned with securing the topology discovery. Therefore it does not guarantee that malicious nodes, which complied with its operation during route discovery, would still comply during the actual data transmission at a later time.

SLSP assumes that each node has a public/private key pair where certification is provided by a certification authority.

A malicious node masquerading as one of its neighbours with the same MAC address can flood the victim's neighbourhood with false link-state updates. Due to SLSP's "duplicate MAC address detection" functionality, other neighbours of the victim will reject all the link-state updates originating from both the malicious node and the victim. Thus the malicious node can succeed in launching a DoS attack against the victim.

Secure Message Transmission (SMT)

SMT (Papadimitratos & Haas, 2003b) protocol aims to safeguard the data transmission against malicious behavior of network nodes. It works in an end-to-end manner. It assumes that there exists a routing protocol that can find a set of routes to each destination.

With a set of routes to a destination, SMT disperses each outgoing message into several pieces based on Rabin's (1989) algorithm. Rabin's algorithm adds limited redundancy to each data packet to allow recovery from a certain number of faults. The message and the redundancy data are divided into a number of packets so that the destination node can reconstruct the whole message even if m out of n packets are received successfully.

SMT requires a security association, similar to SRP, to exchange the required information for successful reconstruction of the dispersed pieces. This requirement obviates the need for a separate secured layer that can perform the security association properly tolerating potential malicious activities.

INTRUSION DETECTION SYSTEMS FOR THE NETWORK LAYER

The security protocols discussed in the previous section have been designed to prevent attacks happening on the first place. In the course of time if any of the nodes in a network is compromised and consequently does not follow protocol specification properly, these protocols may not detect the misbehaving nodes. Though SRP and SLSP contain monitoring systems, these are not robust enough to detect various anomalies. Next we discuss some of the intrusion detection systems designed to detect node misbehaviour in ad-hoc networks.

Watchdog and Pathrater

Marti, Giuli, Lai, and Baker (2000) have proposed an intrusion detection technique, known as Watchdog, to detect nodes that agree to forward packets but fail to do so. They have used another module, known as Pathrater, which uses the information from Watchdog and helps the routing protocol to avoid misbehaving nodes. Watchdog and Pathrater are best implemented on top of source routing protocols, such as DSR.

When a node X forwards a packet, its Watchdog module verifies that the next node Y in the routing path also forwards the packet to another node Z . X 's Watchdog module does this by listening to the transmissions of Y promiscuously. If Y does not forward the packet to Z , X 's Watchdog module increments a failure tally. Once the failure tally exceeds a certain threshold, Y is considered to be misbehaving.

The Pathrater module of each node maintains a rating for every other node it knows about in the network. The Pathrater computes the rating, that is, reliability, of each node based on the information obtained from its Watchdog module. A path metric is calculated averaging the node ratings in the path and is used to pick the path most likely to be reliable among the paths between a source and destination pair.

The Watchdog module cannot detect misbehaving nodes in the presence of: (1) Ambiguous collisions, (2) Receiver collisions, (3) Limited transmission power, (4) Directional antennas, (5) False misbehavior, and (6) Partial dropping. Ambiguous collisions prevent a node from overhearing transmissions of its neighbours. In receiver collisions X 's Watchdog module can tell if Y has forwarded a packet to Z but not if Z has received it or not. Thus if Y wants to circumvent the Watchdog module of X , it can purposefully cause a collision at Z by forwarding the packet to Z when Z is transmitting. A misbehaving node Y can limit its transmission power such that the transmitted signal is strong enough to reach the

previous node X but not the actual recipient Z . A directional antenna prevents neighbouring nodes, which are not within its footprint, from overhearing its transmissions, thus making Watchdog modules ineffective. False misbehavior occurs when a node reports incorrect information about benign nodes. In partial dropping a malicious node can circumvent a Watchdog by dropping packets at a rate lower than the threshold set at that Watchdog.

Techniques for Intrusion Resistant Ad-Hoc Routing Algorithms (TIARA)

Ramanujan et al. (2000) have presented some general techniques, collectively called TIARA, to protect ad-hoc networks from resource depletion, packet dropping and delaying, tunnelling, and replay attacks. TIARA also aims to keep the network operational at an acceptable level during such attacks. It is independent of any specific ad-hoc routing algorithm.

TIARA consists of the following techniques: (1) flow-based route access control (FRAC), (2) flow monitoring, (3) source-initiated route switching, (4) fast authentication, (5) referral based resource allocation, (6) multi-path routing, and (7) incorporating sequence numbers. Table 1 shows the aforementioned design techniques with associated various attacks.

FRAC requires each routing node to maintain an access control rule base that defines the list of authorised flows that the node may forward. The router drops packets belonging to unauthorised flows. The incorporation of FRAC in existing routing protocols requires that flow identifiers instead of destination addresses index the routing tables. Moreover the routing protocol must forward packets based on flow identifiers as opposed to the destination address.

Multi-path routing allows routing nodes to discover and maintain multiple legitimate routes between the source and destination pair corresponding to a data flow. This enables seamless

Table 1. Countermeasure techniques with TIARA for various attacks

Attacks	TIARA countermeasures
Resource depletion	Flow-based route access control, fast authentication, referral-based resource allocation, and sequence numbers
Packet dropping or delaying	Multi-path routing, flow monitoring, source-initiated route switching
Tunnelling	Multi-path routing, flow monitoring, source-initiated route switching

flows of traffic along unharmed routes in the event of attacks on some of the routes.

Source-initiated route switching enables the source to choose a route or a set of routes among all the legitimate routes discovered with a multi-path routing protocol.

A flow monitoring mechanism detects the failure of a route and notifies the source. This provision enables the source-initiated route-switching module to switch the traffic flow to an alternate route and thus maintains seamless flow of traffic.

The flow monitoring system in the source node sends periodic flow status messages to the destination node. The flow status message contains the number of packets associated with the flow that has been transmitted by the source since the last flow status message was sent. This message is encrypted and protected by digital signature to ensure integrity of the message.

The flow monitoring function at the destination node keeps track of the number of packets successfully received from the source between consecutive flow status messages. If the destination does not receive a flow status message for a long time, or the number of packets received is less or more than a predefined threshold, it sends a route failure message to the source along all alternate routes that exist between the two nodes.

Fast authentication is a lightweight mechanism for authenticating data packets at each forwarding

node. It relies on placing a path label, that is, an identification mark, of a packet at a node specific secret location within the packet that may differ for different nodes in the path between the source and destination. The information on the node specific secret location is sent to each node in a secured way during the route establishment process.

Sequence numbers, combined with FRAC and fast authentication, provides a countermeasure for replay attacks. Similar to the fast authentication technique, the source embeds sequence numbers in node specific locations, corresponding to the nodes between the source and destination, within each data packet.

Referral-based resource allocation is claimed to prevent two intruders generating excessive traffic flow between themselves by collusion, hence depleting network resources of the legitimate intermediate nodes between them. In this scheme a node defines an initial threshold that it will allocate for a flow passing through it. Additional resources can only be granted if such a request is presented with referrals from a certain number of trusted nodes. These referrals from the trusted nodes are assumed to be digitally signed by them and have a time-bound validity.

Though the flow monitoring system may identify an attacked route, it cannot find the intruder since the system receives failure messages only from the destination node instead of each intermediate node. Since the thresholds are not

dynamically adapted, a malicious node may fool the detection mechanism.

Intrusion Detection Techniques

Zhang and Lee (2000) and Zhang, Lee, and Huang (2003) have developed a distributed and cooperative IDS for detecting falsifying of route information and random packet dropping (i.e., grey-hole attack) in MANETs.

In the proposed architecture individual IDS agents are placed on each node. Each IDS agent runs independently, detects intrusion from local traces, and initiates a response. If the evidence of an anomaly from a local trace is inconclusive, neighbouring IDS agents cooperate to resolve the issue.

For multi-layer integrated systems, individual IDS agents are placed at each layer on each node. If a node detects a local intrusion at a higher layer, lower layers are notified of the anomaly. The detection module at the lower layer then further investigates the attack, responds by blocking access from the offending nodes, and notifies other nodes of the incident.

Each IDS agent can be structured into six parts: (1) data collection module, (2) local detection engine, (3) cooperative detection module, (4) local response module, (5) global response module, and (6) secure communication module.

The data collection module collects audit traces and activity logs within the radio range of the respective node. The local detection engine uses these data to detect anomalies locally. The cooperative detection module is used whenever the local detection module requires broader data sets or collaboration among neighbouring IDS agents. The local response module triggers actions local to the node. Some of the likely responses are: (1) re-initializing the communication channel between nodes, and (2) identifying compromised nodes and re-organizing the network topology to exclude the compromised node. The global response module is responsible for coordinating

the intrusion response actions of other nodes in the network. A secure communication module provides a secured communication channel among IDS agents.

To detect the aforementioned attacks, the anomaly detection module uses information-theoretic measures (Cover & Thomas, 1991), namely entropy and conditional entropy, to describe the characteristics of normal information flow and uses a classification algorithm to distinguish between normal and abnormal behaviors. The following steps are used for anomaly detection: (1) select or partition audit data into subsets so that each subset contains one class of data (i.e., low entropy or high information gain), (2) perform appropriate data transformation based on the entropy measures, for example, construct new features with low entropy, (3) form classifier using training data, (4) apply the classifier to classify the collected data, and (5) produce intrusion reports.

The authors have suggested using local data sources for anomaly detection since information obtained from external data sources can be compromised and hence cannot be trusted. The information provided by the local data sources are: (1) local routing information including cache entries and traffic statistics, and (2) position location with GPS to provide location and velocity information of nodes.

To evaluate the feasibility of the proposed IDS scheme, the authors have conducted a series of experiments with DSR, AODV, and DSDV in a simulation environment. Results show that the proposed intrusion detection scheme is best suited for on-demand ad-hoc routing protocols (e.g., DSR and AODV) instead of table driven protocols (e.g., DSDV).

The random packet dropping detection scheme relies on overhearing transmissions by listening in promiscuous mode. However, such dependence has short shortcomings as stated in the section, *Watchdog and Pathrater*. Collaboration with other nodes can be achieved properly if a secured

communication layer acts as a shield to protect legitimate nodes from intruders.

Security Enhancements in the AODV Protocol

Bhargava and Agrawal (2001) have extended the Watchdog and Pathrater described previously to enhance the security in the AODV routing protocol. Their protocol consists of an intrusion detection model (IDM) and an intrusion response model (IRM).

In this scheme each node employs an IDM that uses neighbourhood information to detect misbehavior of its neighbours. When the misbehavior count (MalCount) for a particular neighbour reaches a predefined threshold, the information is sent to other nodes. When a node receives this information, it checks the local MalCount for the malicious node and adds its result to the initiator's response.

Each node also uses an IRM that identifies that another node has been compromised if the local MalCount corresponding to the malicious node increases beyond a threshold value. In such a case, the IRM module propagates this information to the entire network by transmitting a special type of packet called a Mal packet. If a node receiving the Mal packet also suspects the node indicated in the Mal packet to be malicious, it reports its suspicion to the entire network by transmitting a ReMal packet. If two or more nodes report about a particular node misbehaving, a special type of packet called the purge packet is transmitted to isolate the malicious node from the network. All nodes, that have routes through the identified compromised node, look for newer routes. All packets received from the compromised nodes are dropped as well.

The proposed scheme is claimed to identify false route request, DoS (denial of service), compromise of a destination and impersonation attacks in the following ways:

1. **Distributed False Route Request:** A malicious node may disrupt the normal operation of the network by sending frequent unnecessary route requests. If the nodes receive a number of route requests for a particular source and destination pair greater than a threshold count within a specific time interval, the source is identified as malicious.
2. **DoS:** A malicious node can transmit false control and data packets at an excessive rate and thus launch DoS attacks against legitimate nodes so that they cannot access the network resources (e.g., bandwidth) properly. This type of attack can be identified if a node generates packets at higher frequency than a threshold.
3. **Destination is Compromised:** This type of attack is detected if a source does not receive a reply from a particular destination within a given time interval. Furthermore, the neighbours generate probe or HELLO packets to determine connectivity. In this model if a node does not respond to a route request, it is identified as a malicious node.
4. **Impersonation:** A sender is identified as a malicious node if the receiver cannot decrypt the received packet sent by that sender.

If the threshold values are not adapted dynamically, a malicious node may be clever enough to work around the proposed detection system. The authors have not provided any mechanism to adjust the threshold values dynamically. Moreover it is not clear how a node can monitor ongoing traffic of its neighbours if the neighbours transmit packets with limited transmission power and directional antennas.

COoperative of Nodes, Fairness In Dynamic Ad-hoc NeTworks (CONFIDANT)

CONFIDANT (Buehger & Boudec, 2002a, 2002b) is an extended version of Watchdog and

Pathrater. Unlike Watchdog and Pathrater, this scheme punishes the misbehaving nodes. Nodes with bad reputations are isolated in order to limit their activity in the network. Thus CONFIDANT, unlike Watchdog and Pathrater, stimulates misbehaving nodes to contribute to the normal operations of the network in order to be able to get services from other nodes.

CONFIDANT consists of four major components: (1) monitor, (2) reputation system, (3) path manager, and (4) trust manager. Each node is assumed to be equipped with all these components.

The monitor observes any routing or forwarding misbehavior of its neighbouring nodes by listening to the transmission channel in promiscuous mode. It then passes this information to the reputation system for further analysis.

The reputation system manages a table consisting of entries for nodes and their ratings. The rating of a node is updated when there is sufficient evidence that the node in question has misbehaved significantly. The rating is changed according to a rating function that assigns different weights to the type of behavior detection, namely the greatest weight for own experience, a smaller weight for observed activities and smallest weight for reported misconduct. There is a time limit for the reputation information in order to lessen the effect of false reporting.

The path manager takes input from the reputation system and performs the following functions: (1) path re-ranking according to the level of reputation of associated intermediate nodes, (2) deletion of paths containing malicious nodes, (3) actions on receiving a request for a route from a malicious node (e.g., ignore the request), and (4) actions on receiving a request for a route containing a malicious node from a benign node (e.g., alert the benign node).

The trust manager of a node issues ALARM messages to the node's friends. ALARM messages are generated if the node experiences, observes, or receives a report of malicious behavior. Upon

receiving an ALARM message from a node, the trust manager assesses the trustworthiness of the received message according to the trust level of the reporting node.

CONFIDANT uses a mechanism similar to PGP (pretty good privacy) for expressing various levels of trust (e.g., unknown, none, marginal, and complete), key validation, and certification. CONFIDANT calculates the validity of a public key by examining the trust levels of all attached signatures and computes a weighted score of the validity. For example, two marginally trusted signatures could be accepted as a completely trusted signature.

If the trust manager considers the received report as trusted and has received similar trusted reports from other nodes, it passes this information to the reputation system so that the perception about the allegedly malicious node can be updated.

Since misbehaving nodes are isolated, reputable nodes may suffer from congestion as most of the routes are likely to pass through them. CONFIDANT allows negative ratings from other nodes, which may result in false accusation. Moreover, like Watchdog and Pathrater, CONFIDANT may not work properly in networks where nodes use limited transmission power and directional antennas.

Collaborative REputation (CORE)

CORE (Michiardi & Molva, 2002a) is a reputation-based system similar to CONFIDANT. Unlike CONFIDANT, it does not allow negative ratings to be broadcast by other nodes. This prevents false accusation. CORE is a generic mechanism that can be integrated with several network and application layer functions. Examples of the network functions include performing route discovery, forwarding data packets, and so forth.

A CORE consists of a set of reputation tables (RT) and a Watchdog (WD) module. There is one RT for each function that has to be monitored and

a global RT that combines the different values of reputation calculated for different functions. CORE defines two types of reputations, namely subjective and indirect, each of which is calculated with respect to different functions. A subjective reputation is calculated directly from a node's own observation, and can take both negative and positive values. An indirect reputation is evaluated considering the observations of other nodes, and can take only positive values. Allowing only positive values for indirect reputation prevents DoS attacks based on malicious broadcasting of negative ratings for legitimate nodes.

The general formula to calculate subjective reputation is:

$$r_{s_i}(s_j | f) = \sum \rho(t, t_k) \cdot \sigma_k$$

Here $r_{s_i}(s_j | f) \in [-1, 1]$ denotes the subjective reputation of node s_j calculated by its neighbour s_i at time t with respect to the function f . $\sigma_k \in [-1, 1]$ is a rating factor given to the k^{th} observation taken at time t_k and $\rho(t, t_k)$ is a time dependent function that gives higher relevance to past values of σ_k .

The indirect reputation is denoted by $ir_{s_i}(s_j | f)$. It can be calculated using a similar expression to that used for $r_{s_i}(s_j | f)$ provided that the corresponding rating factor takes only positive values. Now the aforementioned reputation information can be combined with the following formula:

$$r_{s_i} = \sum w_k \{r_{s_i}(s_j | f) + ir_{s_i}(s_j | f)\} \quad (1)$$

Here w_k represents the weight associated with the functional reputation value. For example packet forwarding can be given more weight than the routing function since the former has a more important impact on the overall performance compared to the latter (Michiardi & Molva, 2002b).

Every time a node needs to monitor the correct execution of a function by a neighbouring node, it triggers a WD specific to that function. The WD compares the observed result with an expected value to detect if the monitored function has been

executed properly. If the WD detects any anomaly, a negative value is assigned to the rating factor corresponding to the monitored function. With this information a new reputation for the malicious neighbour is calculated using Equation (1).

In CORE the most reputed nodes may become congested as most of the routes are likely to pass through them. As in Watchdog and Pathrater, the limitations of the monitoring system in networks with limited transmission power and directional antennas have not been addressed.

AODV State Transition Analysis Technique (AODVSTAT)

AODVSTAT (Vigna, Gwalani, Srinivasan, Belding-Royer, & Kemmerer, 2004) is a state transition analysis technique (STAT) (Ilgun, Kemmerer, & Porras, 1995) based IDS designed for detecting attacks against AODV. The proposed system tries to reduce the number of false positives.

STAT was initially designed to model host-based and network-based intrusions in wired networks. In state transition analysis, computer penetrations (i.e., attacks) are described as a sequence of actions that leads an attacker from some initial state on a system to a target compromised state. Between the initial and the compromised states there are one or more intermediate state transitions that an attacker has to perform to achieve the desired compromised state.

When the initial and compromised states have been identified, the key actions (or signature actions) are determined. Here signature actions refer to those actions that would prevent an attack from completing successfully if one or more of those actions are omitted from the execution of an attack scenario. STAT can be a useful approach for intrusion analysis since it requires the analyst to identify a minimum number of signature actions for an attack and to organise them visually similar to a state transition diagram. In AODVSTAT the signature actions are either data or AODV control packets exchanged over a wireless network.

AODVSTAT sensors are deployed on a subset of nodes where each sensor performs state analysis of the packet streams to detect signs of intrusion. Figure 4 shows the architecture of an AODVSTAT sensor.

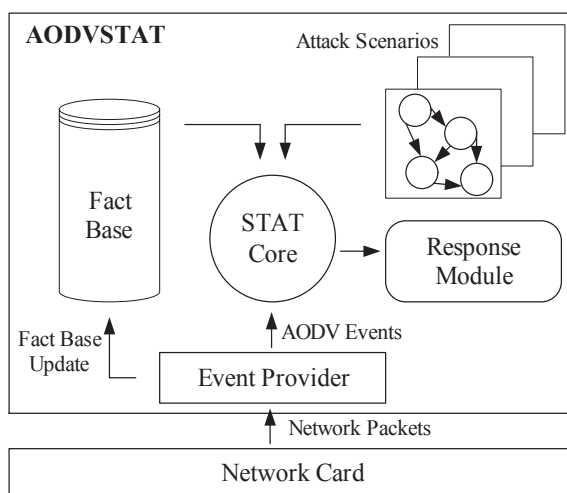
The detection process relies on an internal fact-base which contains information about neighbouring nodes. The internal fact-base is updated by analysing the observed data and AODV control packets transmitted from the neighbouring nodes. In the distributed mode, the fact base also contains information from other nodes obtained by means of UPDATE messages.

Each sensor monitors surrounding network traffic. The transmitted data packets are monitored to determine how much traffic has been generated, received, and forwarded by each node. On the other hand the control packets are monitored to extract the AODV sequence numbers of active nodes, the IEEE 802.11 header details and the MAC/IP pairs of the nodes residing within the range of the sensor. The packets retrieved are then matched against a number of state transition attack scenarios. When a match is found, an intrusion alert is initiated.

Here are some of the attacks that AODVSTAT can detect:

1. **Impersonation:** Many routing protocols use MAC/IP address pairs to uniquely identify a node. To detect impersonation attacks when a packet arrives with a new MAC/IP address pair, the pair is stored for the first time. If another packet arrives with the same IP but different MAC, or vice versa, the packet is detected as spoofed.
2. **Dropping of Packets:** To detect dropping of RREQ (route request) packets, the IDS maintains a list of all its neighbours. This list is refreshed dynamically to allow changes in network topology. The IDS detects a dropped RREQ packet if its neighbour fails to reply to that packet with another RREQ or RREP (route reply) packet. To detect dropped RREP packets, the IDS checks whether its neighbour forwards the received RREP packet towards the source node, provided that the neighbour is not the intended final destination for the RREP packet. Detecting

Figure 4. An AODVSTAT architecture



dropping of DATA packets is similar to the scenario for dropping RREP packets.

3. **Resource Depletion Attack:** To detect this attack the IDS maintains a count of the number of packets it receives from each node within a specific time window. If the count exceeds a certain threshold, then an alert is signaled.
4. **False Propagation of High Sequence Number:** This kind of attack can be detected by using a distributed set of sensor nodes that send neighbour information to every other sensor node. The sensor nodes as a group check if an RREP packet with a sequence number for a particular destination is received with a value greater than a certain threshold. If so the attack is identified.
5. **Diverting Traffic through Malicious Node:** It can be detected by a combination of impersonation and the packet drop attack.

The proposed scheme does not consider compromised nodes sending false intrusion information to other nodes. Moreover, like Watchdog and Pathrater AODVSTAT does not provide any solution for networks where nodes can transmit packets with limited transmission power, different frequency channels, or directional antennas.

TWOACK

Balakrishnan, Deng, and Varhney (2005) have proposed a different way to detect packet dropping in ad-hoc networks that addresses the problems of receiver collisions, limited transmission power, and directional antennas discussed in Watchdog and Pathrater. The scheme can be added on to any source routing protocol such as DSR.

Suppose node A has discovered a route to F with a source route $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$. In TWOACK when B forwards a packet for A , C (the node two hops away from A) receives the packet and sends an acknowledgement to A indicating B has forwarded the packet properly. If

A does not get an acknowledgment for the packet within a certain timeout period it suspects B to be misbehaving. The same procedure is carried out by every set of three consecutive nodes along the source route.

In TWOACK each forwarded packet has to be acknowledged which may contribute to traffic congestion on the routing path. S-TWOACK (Selective TWOACK) reduces this extra traffic by sending a single acknowledgment for a certain number of packets instead of a single packet.

The proposed scheme may fail if two consecutive nodes along the route collude to misbehave. SAHN-IDS (Islam, Pose, & Kopp, 2005b) provides a mechanism to minimise this problem. Moreover, TWOACK/S-TWOACK cannot detect the misbehavior of a forwarding node if it delays packet transmissions for selective nodes.

OPEN RESEARCH CHALLENGES

Multimedia applications, such as video conferencing and interactive gaming, often have high bandwidth requirements and sometimes need very fast response times. Running such demanding applications over ad-hoc networks requires the network to be able to provide high bandwidth and real-time response. Security protocols often use computationally expensive encryption and authentication mechanisms, and need to exchange many messages for their correct operation. This computational and communications overhead make deploying secure multimedia applications on ad-hoc networks very challenging.

As the size and density of ad-hoc networks grow, the communication overhead of the security protocols increases. This can have an adverse effect of network performance in ad-hoc networks having limited bandwidth. In some cases it may be impractical to provide the required security, with its attendant overheads, since doing so may compromise the essential performance requirements of the multimedia applications.

Some ad-hoc networks, for example, sensor networks, consist of nodes having limited battery power and computational resources. Security protocols with high computational overheads, for example, computations for RSA, may not be feasible for this type of network.

There are tradeoffs among security strength, communication overhead, computational complexity, energy consumption, and scalability. Further security and performance analyses can be used to find a balance between security protocols and other network constraints so that the overall security strength and network performance are optimised to suit the requirements of network operations and users.

The security protocols discussed so far are mostly implemented in a single layer (except CORE) in the network protocol stack. One of the disadvantages of having a single layered security approach is if the security protocol in the single layer fails, other layers will be exposed to malicious attacks. Therefore a multi-layered security approach is required that can be integrated into possibly every component in the network protocol stack. The performance of the network should not critically depend on any single security component so that the overall security system is effective despite the breakdown of one or more security components.

The existing security protocols are designed with certain attacks in mind and hence may not work well in the presence of unanticipated attacks. One of the main reasons for this limitation is that the possible attacks are not well modelled. A comprehensive model of possible attacks needs to be developed in order to design robust security protocols that can handle a bigger problem space.

Malicious attacks and network faults due to node misconfigurations, extreme network overload, or operational failures share common symptoms (Yang et al., 2004). Security protocols should be able to cope with all types of anomalies.

To strengthen the level of security, a security protocol should have an access control management facility so that different nodes can be given different privileges to access the network services. SAR incorporates a trust of hierarchy in the network that provides a simple access control management in the network. Other concepts, such as the password-capability (Anderson, Pose, & Wallace, 1986; Castro, 1996; Kopp, 1997; Pose, 2001), can be incorporated in a security protocol to provide a fine grained access control management facility in the network. Islam, Pose, and Kopp (2004, 2005a, 2005c) have proposed a security protocol called SSP (SAHN security protocol) based on password-capability for static ad-hoc networks.

CONCLUSION

In this chapter we examined a number of possible classes of attacks on ad-hoc networks. These included drop packets, delay transmission, protocol field modification, message fabrication, replay attack, broadcast storm attack, piggyback attack, tunnelling attack, and identity theft.

For these attacks we outlined preventive strategies for dealing with them and discussed the merits of these approaches. SAR provides various alternatives to authenticate and encrypt routing packets. It incorporates a trust hierarchy in the network in order to associate different privileges or “trust levels” with different nodes. SRP ensures that each pair of end nodes can differentiate between legitimate route requests and route replies. SEAD attempts to protect the route updates of DSDV routing protocol from malicious attacks using one-way hash chains. Ariadne authenticates routing messages using any of the three schemes: (1) shared secrets between each pair of nodes, (2) shared secrets between communicating nodes combined with broadcast authentication called TESLA, or (3) digital signature. ARAN uses public key cryptography for authenticating route

request, reply and error packets of AODV routing protocol. SAODV uses one-way hash chains to authenticate the hop count field of route request and route reply packets, and digital signature to sign all fields of route error, and most of the fields (except for the hop count field) of route request and route reply packets at each intermediate node for AODV routing protocol. Packet leases handle the tunnelling attack by adding additional information in each packet to determine if the packet has traversed an unrealistic distance. SLSP secures the discovery of neighbours and distribution of link-state information of table driven or pro-active routing protocols using digital signatures and one-way hash chains. SMT aims to safeguard the data transmission against malicious behaviour of network nodes on end-to-end basis. All these protocols have been designed to plug various forms of attack, hence making the system more robust.

We also examined mechanisms and security protocols that can deal with various forms of anomalous behaviour of ad-hoc networks. Watchdog and Pathrater detect nodes that agree to forward packets but fail to do so in a DSR-like routing protocol and help the routing protocol to avoid misbehaving nodes. TIARA consists of some general techniques to protect ad-hoc networks from resource depletion, packet dropping and delaying, tunnelling, and replay attacks. The scheme proposed by Zhang et al. (2000, 2003) is a distributed and cooperative IDS that detects falsifying route information and random packet dropping. Bhargava and Agrawal (2001) have extended Watchdog and Pathrater to enhance the security in the AODV routing protocol. CONFIDANT is an extension of Watchdog and Pathrater that punishes the misbehaving nodes by limiting their activity in the network. Thus CONFIDANT encourages misbehaving nodes to contribute to the normal operations of the network in order to be able to get services from other nodes. CORE is a generic reputation-based IDS that can be integrated with several network and application

layer functions. AODVSTAT is a state transition analysis technique (STAT) based IDS designed for detecting attacks against AODV. TWOACK detects packet dropping in ad-hoc networks using DSR like routing protocols that addresses the problems of receiver collisions, limited transmission power, and directional antennas of Watchdog and Pathrater. All of these protocols deal with anomalous network behaviour and attempt to limit its spread and indeed preferably to isolate the offending network nodes. In so doing they assist in preserving the security of the system.

Our study of various security protocols suggests that no particular protocol or class of protocols is best suited for all scenarios. The field of security in ad-hoc networks is still an open research challenge, however significant security protocol developments as outlined in this chapter have led the way towards the elusive goal of a demonstrably secure system.

REFERENCES

- Anderson, M., Pose, R., & Wallace, C. S. (1986). A password-capability system. *The Computer Journal*, 29(1), 1-8.
- Balakrishnan, K., Deng, J., & Varhney, P. K. (2005). TWOACK: Preventing selfishness in mobile ad hoc networks. *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 4 (pp. 2137-2142).
- Bhargava S., & Agrawal, D. P. (2001). Security enhancements in AODV protocol for wireless ad-hoc networks. *Proceedings of the Vehicular Technology Conference*, 4 (pp. 2143-2147).
- Buchegger, S., & Boudec, J.-Y. L. (2002a). Nodes bearing grudges: Towards routing security, fairness and robustness in mobile ad hoc networks. *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing* (pp. 403-410).

- Buchegger, S., & Boudec, J.-Y. L. (2002b). Performance analysis of the CONFIDANT protocol. *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ISBN 1-58113-501-7 (pp. 455-465).
- Capkun, S., Buttyan, L., & Hubaux, J.-P. (2003). Self-organizing public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1), 52-64.
- Castro, M. D. (1996). *The walnut kernel: A password-capability based operating system*. Ph.D. Thesis, Monash University, Clayton, Australia.
- Coppersmith, D., & Jakobsson, M. (2002). Almost optimal hash sequence traversal. *Conference on Financial Cryptography (FC)*, Lecture Notes in Computer Science (pp. 102-119).
- Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. Wiley.
- Douceur, J. (2002). The Sybil attack. *1st International Workshop Peer-to-Peer Systems (IPTPS)*, Lecture Notes in Computer Science, ISBN 3-540-44179-4, 2429 (pp. 251-260).
- Gupte, S., & Singhal, M. (2003). Secure routing in mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1), 151-174.
- Hu, Y.-C., Johnson, D. B., & Perrig, A. (2002a). SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, ISBN 0-7695-1647-5 (pp. 3-13).
- Hu, Y.-C., Perrig, A., & Johnson, D. B. (2002b). Ariadne: A secure on-demand routing protocol for ad-hoc networks. *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-486-X (pp. 12-23).
- Hu, Y.-C., Perrig, A., & Johnson, D. B. (2003). Packet leases: A defense against wormhole attacks in wireless ad hoc networks. *Proceedings of the ACM Workshop on Wireless Security (WiSe)* (pp. 30-40).
- Ilgun, K., Kemmerer, R. A., & Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3), 181-199.
- Islam, M. M., Pose, R., & Kopp, C. (2004). A link layer security protocol for suburban ad-hoc networks. *Australian Telecommunication Networks and Applications Conference (ATNAC)* (pp. 174-177).
- Islam, M. M., Pose, R., & Kopp, C. (2005a). Link layer security for SAHN protocols. *3rd IEEE PerCom Workshops on PWN* (pp. 279-283).
- Islam, M. M., Pose, R., & Kopp, C. (2005b). An intrusion detection system for suburban ad-hoc networks. *IEEE Tencn*.
- Islam, M. M., Pose, R., & Kopp, C. (2005c). Suburban ad-hoc networks in information warfare. *6th Australian InfoWar Conference*, Geelong, Australia.
- Johnson, D., & Maltz, D. (1996). Dynamic source routing in ad hoc wireless networks. In T. Imielinski & H. Korth (Eds.), *Mobile computing* (vol. 353, pp. 151-181). Kluwer Academic Publishers.
- Kopp, C. (1997). *An I/O and stream inter-process communications library for a password capability system*. Masters Thesis, Monash University, Clayton, Australia.
- Kopp, C., & Pose, R. (1998). Bypassing the home computing bottleneck: The suburban area network. *3rd Australasian Computer Architecture Conference (ACAC)*, 20(4), 87-100. Springer-Verlag, ISBN: 981-3083-93-X.

- Luo, H., & Lu, S. (2000). *Ubiquitous and robust authentication services for ad hoc wireless networks*. Technical Report, TR-200030, Department of Computer Science, UCLA.
- Marti, S., Giuli, T. J., Lai, K., & Baker, M. (2000). Mitigating routing misbehavior in mobile ad-hoc networks. *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-197-6 (pp. 255-265).
- Michiardi, P., & Molva, R. (2002a). CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, ISBN 1-4020-7206-6 (pp. 107-121).
- Michiardi, P., & Molva, R. (2002b). Simulation-based analysis of security exposures in mobile ad-hoc networks. *Proceedings of the European Wireless Conference*.
- Papadimitratos, P., & Haas, Z. J. (2002). Secure routing for mobile ad hoc networks. *Proceedings of the SCS Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS)*.
- Papadimitratos, P., & Haas, Z. J. (2003a). Secure link state routing for mobile ad hoc networks. *Proceedings of the Symposium on Applications and the Internet Workshops (SAINT)* (pp. 27-31).
- Papadimitratos, P., & Haas, Z. J. (2003b). Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1), 193-209.
- Perkins, C., & Bhagwat, P. (1994). Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM)* (pp. 234-244).
- Perkins, C. E., & Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)* (pp. 90-100).
- Perrig, A., Canetti, R., Tygar, J. D., & Song, D. (2000). Efficient authentication and signing of multicast streams over lossy channels. *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 56-73).
- Pose, R. (2001). Password-capabilities: Their evolution from the password-capability system into walnut and beyond. In G. Heiser (Ed.), *6th Australasian Computer Systems Architecture Conference (ACSAC)*, 23 (pp. 105-113).
- Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of ACM*, 36(2), 335-348.
- Ramanujan, R., Ahamad, A., Bonney, J., Haggelstrom, R., & Thurber, K. (2000). Techniques for intrusion-resistant ad hoc routing algorithms (TIARA). *Proceedings of the 21st Century Military Communications Conference (MILCOM)*, ISBN 0-7803-6521-6 (pp. 660-664).
- Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communication ACM*, 21(2), 120-126.
- Sanzgiri, K., Dahill, B., Levine, B. N., Shields, C., & Belding-Royer, E. M. (2002). A secure routing protocol for ad hoc networks. *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)* (pp. 78-89).
- Sanzgiri, K., Dahill, B., Levine, B. N., Shields, C., & Belding-Royer, E. M. (2005). Authenticated routing for ad hoc networks. *IEEE Journals on Selected Areas in Communications*, 23(3), 598-610.

Security in Ad-Hoc Networks

- Schnorr, C. P. (1991). Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3), 161-174.
- Shamir, A. (1979). How to share a secret. *Communications of ACM*, 22(11), 612-613.
- Vigna, G., Gwalani, S., Srinivasan, K., Belding-Royer, E. M., & Kemmerer, R. A. (2004). An intrusion detection tool for AODV-based ad hoc wireless networks. *Proceedings of the Annual Computer Security Applications Conference (ACSAC)* (pp. 16-27).
- Yang, H., Luo, H., Ye, F., Lu, S., & Zhang, L. (2004). Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, 11(1), 38-47.
- Yi, S., & Kravets, R. (2002). *Key management for heterogeneous ad hoc wireless networks*. Technical Report, UIUCDCS-R-2002-2290, Department of Computer Science, University of Illinois at Urbana-Champaign.
- Yi, S., Naldurg, P., & Kravets, R. (2001). Security-aware ad hoc routing for wireless networks. *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, ISBN 1-58113-428-2 (pp. 299-302).
- Zapata, M. G., & Asokan, N. (2002). Securing ad hoc routing protocols. *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe)*, ISBN 1-58113-585-8 (pp. 1-10).
- Zhang, Y., & Lee, W. (2000). Intrusion detection in wireless ad-hoc networks. *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, ISBN 1-58113-197-6 (pp. 275-283).
- Zhang, Y., Lee, W., & Huang, Y. (2003). Intrusion detection techniques for mobile wireless networks. *Kluwer Wireless Networks*, 9(5), 545-556.
- Zhou, L., & Haas, Z. J. (1999). Securing ad hoc networks. *IEEE Networks*, 13(6), 24-30.