Ad hoc networks security

Pietro Michiardi and Refik Molva

1. Introduction

An *ad hoc* network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. In such an environment, it may be necessary for one mobile host to enlist the aid of other hosts in forwarding a packet to its destination, due to the limited range of each mobile host's wireless transmissions. Mobile ad hoc networks (MANET) do not rely on any fixed infrastructure but communicate in a self-organized way.

Security in MANET is an essential component for basic network functions like packet forwarding and routing: network operation can be easily jeopardized if countermeasures are not embedded into basic network functions at the early stages of their design. Unlike networks using dedicated nodes to support basic functions like packet forwarding, routing, and network management, in ad hoc networks those functions are carried out by all available nodes. This very difference is at the core of the security problems that are specific to ad hoc networks. As opposed to dedicated nodes of a classical network, the nodes of an ad hoc network cannot be trusted for the correct execution of critical network functions.

If *a priori trust relationship* exists between the nodes of an ad hoc network, entity authentication can be sufficient to assure the correct execution of critical network functions. A priori trust can only exist in a few special scenarios like military networks and corporate networks, where a common, trusted authority manage the network, and requires tamper-proof hardware for the implementation of critical functions. Entity authentication in a large network on the other hand raises key management requirements. An environment where a common, trusted authority exists, is called a *managed environment*.

When tamper-proof hardware and strong authentication infrastructure are not available, like for example in an *open environment* where a common authority that regulates the network does not exist, any node of an ad hoc network can endanger the reliability of basic functions like routing. The correct operation of the network requires not only the correct execution of critical network functions by each participating node but it also requires that each node performs a fair share of the functions. The latter requirement seems to be a strong limitation for wireless mobile nodes whereby power saving is a major concern. The threats considered in the MANET scenario are thus not limited to maliciousness and a new type of misbehavior called selfishness should also be taken into account to prevent nodes that simply do not cooperate.

With *lack of a priori trust*, classical network security mechanisms based on authentication and access control cannot cope with selfishness and cooperative security schemes seem to offer the only reasonable solution. In a cooperative security scheme,

node misbehavior can be detected through the collaboration between a number of nodes assuming that a majority of nodes do not misbehave.

The rest of the chapter is organized as follows: section 2 presents the recent research that has been done in order to come up with secure routing protocols for ad hoc networks that cope with threats that are specific to the ad hoc environment. All of the presented secure protocols, however, do not take into account the node selfishness problem, which is detailed in section 3. Recent solutions to combat the lack of node cooperation are presented in section 3. The basic requirement of a large number of proposed security scheme is the presence of a key distribution mechanism managed by a trusted authority that take part in the initialization phase of the network: recent advances in order to provide an automated key management scheme that does not require the presence of any external infrastructure nor bootstrap phase where keys are distributed are presented in section 4. In section 5, currently available security mechanisms implemented in the data link layer are detailed and analyzed. Furthermore section 5.3 focuses on a discussion about the relevance for the ad hoc environment of security mechanisms implemented in the data link layer.

2. Secure Routing

Routing protocols for ad hoc networks are challenging to design: wired networks protocols (such as BGP) are not suitable for an environment where node mobility and network topology rapidly change; such protocols also have high communication overhead because they send periodic routing messages even when the network is not changing. So far, researchers in ad hoc networking have studied the routing problem in a non-adversarial network setting, assuming a reasonably trusted environment. However, unlike networks using dedicated nodes to support basic functions like packet forwarding, routing, and network management, in ad hoc networks, those functions are carried out by all available nodes. This very difference is at the core of the increased sensitivity to node misbehavior in ad hoc networks and the current proposed routing protocols are exposed to many different types of attacks.

Section 2.1 presents and classifies the threats that a misbehaving node can perpetrate to jeopardize the network operation. Recent research brought up the need to take into account node misbehavior at the early stages of the routing protocol design: current efforts in secure routing protocol design are outlined and analyzed in section 2.2.

2.1 Exploits allowed by existing routing protocols

Current ad hoc routing protocols are basically exposed to two different types of attacks: *active* attacks and *passive* attacks. An attack is considered to be active when the misbehaving node has to bear some energy costs in order to perform the threat while passive attacks are mainly due to lack of cooperation with the purpose of saving energy selfishly. Nodes that perform active attacks with the aim of damaging other nodes by causing network outage are considered to be *malicious* while nodes that perform passive

attacks with the aim of saving battery life for their own communications are considered to be *selfish*.

Malicious nodes can disrupt the correct functioning of a routing protocol by *modifying* routing information, by *fabricating* false routing information and by *impersonating* other nodes. Recent research studies [10] brought up also a new type of attack that goes under the name of *wormhole* attack. On the other side, selfish nodes can severely degrade network performances and eventually partition the network (ref EW2002) by simply not participating to the network operation.

2.1.1 Threats using modification

Existing routing protocols assume that nodes do not alter the protocol fields of messages passed among nodes. Malicious nodes can easily cause traffic subversion and denial of service (DoS) by simply altering these fields: such attacks compromise the *integrity* of routing computations. By modifying routing information an attacker can cause network traffic to be dropped, redirected to a different destination or take a longer route to the destination increasing communication delays.

2.1.2 Threats using impersonation

Since current ad hoc routing protocols do not *authenticate* routing packets a malicious node can launch many attacks in a network by masquerading as another node (*spoofing*). Spoofing occurs when a malicious node misrepresents its identity in order to alter the vision of the network topology that a benign node can gather. As an example, a spoofing attack allows to create loops in routing information collected by a node with the result of partitioning the network.

2.1.3 Threats using fabrication

The notation “fabrication” is used when referring to attacks performed by generating false routing messages. Such kind of attacks can be difficult to identify as they come as valid routing constructs, especially in the case of fabricated routing error messages claiming that a neighbor can no longer be contacted.

2.1.4 Wormhole attack

A more subtle type of active attack is the creation of a tunnel (or wormhole) in the network between two colluding malicious nodes linked through a private network connection. This exploit allows a node to short-circuit the normal flow of routing messages creating a virtual vertex cut in the network that is controlled by the two colluding attackers.

2.1.5 Lack of cooperation

A selfish node that wants to save battery life for its own communication can endanger the correct network operation by simply not participating to the routing protocol or by not executing the packet forwarding (this attack is also known as the black hole attack) . Current ad hoc routing protocols can not cope with the selfishness problem and network performances severely degrade.

2.2 Secure routing protocols

Current efforts towards the design of secure routing protocols are mainly oriented to reactive (on-demand) routing protocols such as DSR [12] or AODV [13], where a node attempts to discover a route to some destination only when it has a packet to send to that destination. On-demand routing protocols have been demonstrated to perform better with significantly lower overheads than proactive routing protocols in many scenarios (references) since they are able to react quickly to topology changes, yet being able to reduce routing overhead in periods or areas of the network in which changes are less frequent. It is possible to find, however, interesting security solutions for proactive routing protocols which are worthwhile to mention.

Common to the secure routing protocols proposed in the literature, is the type of attack they address: major effort is put into finding countermeasures against *active attacks* performed by malicious nodes that aim at intentionally disrupt the routing protocol execution while the selfishness problem is not addressed. Furthermore, the prerequisite for all the available solutions is a *managed* environment: in such scenario, nodes whishing to communicate may be able to exchange initialization parameters beforehand, for example within the security of a dedicated network where session keys may be distributed or through a trusted third party.

Following, the major secure routing protocols for ad hoc networks will be outlined and analyzed.

2.2.1 SRP

The Secure Routing Protocol [1] proposed by Papadimitratos and Haas is conceived as an extension that can be applied to a multitude of existing *reactive* routing protocols. SRP combats attacks that disrupt the route discovery process and guarantees the acquisition of correct topological information: a node initiating a route discovery is able to identify and discard replies providing false routing information or avoid receiving them.

The underlying assumption is the existence of a *security association* (SA) between the source node (S) and the destination node (T). The trust relationship could be instantiated, for example, by the knowledge of the public key of the other communicating end. The two nodes can negotiate a shared secret key ($K_{S,T}$) and then, using the SA, verify that the principal that participated in the exchange was indeed the trusted node.

SRP copes with non-colluding *malicious* nodes that are able to modify (corrupt), replay and fabricate routing packets. Based on the dynamic source routing protocol (DSR, references) SRP requires the addition of a 6-word header containing unique identifiers that tag the discovery process and a message authentication code (MAC). In order to initiate a route request (RREQ) the source node has to generate a MAC by a keyed hash algorithm that accepts as input the entire IP header, the basis protocol RREQ packet and the shared key $K_{S,T}$.

The intermediate nodes that relays the RREQ towards the destination measure the frequencies of queries received from their neighbors in order to regulate the query propagation process: each node maintains a priority ranking that is inversely proportional to the queries rate. A node that maliciously pollutes network traffic with unsolicited RREQ will be served last (if not ignored) because of its low priority ranking.

Upon reception of a RREQ, the destination node verifies the *integrity* and *authenticity* of the RREQ by calculating the keyed hash of the request fields and comparing them with the MAC contained in the SRP header. If the RREQ is valid, the destination initiates a route replay (RREP) using the SRP header the same way the source did when initiating the request. The source node discards replays that do not match with pending query identifiers and checks the integrity using the MAC generated by the destination.

The basic version of SRP suffers from the route cache poisoning attack: routing information gathered by nodes that operate in promiscuous mode in order to improve the efficiency of the DSR protocol could be invalid, because fabricated by malicious nodes. The authors propose two alternative designs of SRP that uses an Intermediate Node Reply Token (INRT). INRT allows intermediate nodes that belong to the same group that share a common key (K_G) to validate RREQ and provide valid RREP messages.

SRP suffers also from the lack of a validation of route maintenance messages: route errors packets are not verified. However, in order to minimize the effects of fabricated error messages, SRP source-routes error packets along the prefix of the route reported as broken: as a consequence the source node can verify that the provided route error feedback refers to the actual route and is not generated by a node that is not even part of the route. A malicious node can harm only the route it belongs.

Assuming that the neighbor discovery mechanism maintains information on the binding of the medium access control and the IP addresses of nodes, SRP is proven to be essentially immune to IP spoofing [1].

SRP is, however, not immune to the wormhole attack: two colluding malicious nodes can misroute the routing packets on a private network connection and alter the network topology vision a benign node can collect.

2.2.2 ARIADNE

Hu, Perrig and Johnson present an *on-demand* secure ad hoc routing protocol based on DSR that withstands node compromise and relies only on highly efficient *symmetric* cryptography. ARIADNE guarantees that the target node of a route discovery process can authenticate the initiator, that the initiator can authenticate each intermediate node on the path to the destination present in the RREP message and that no intermediate node can remove a previous node in the node list in the RREQ or RREP messages.

As for the SRP protocol, ARIADNE needs some mechanism to bootstrap authentic keys required by the protocol. In particular, each node needs a shared secret key ($K_{S,D}$, is the shared key between a source S and a destination D) with each node it communicates with at a higher layer, an authentic TESLA [3, 4] key for each node in the network and an authentic "Route Discovery chain" element for each node for which this node will forward RREQ messages.

ARIADNE provides point-to-point *authentication* of a routing message using a message authentication code (MAC) and a shared key between the two parties. However, for authentication of a broadcast packet such as RREQ, ARIADNE uses the TESLA broadcast authentication protocol. ARIADNE copes with attacks performed by *malicious* nodes that modify and fabricate routing information, with attacks using impersonation and, in an advanced version, with the wormhole attack. Selfish nodes are not taken into account.

In ARIADNE, the basic RREQ mechanism is enriched with eight fields used to provide authentication and integrity to the routing protocol:

<ROUTE REQUEST, initiator, target, id, time interval, hash chain, node list, MAC list>

The initiator and target are set to the address of the initiator and target nodes, respectively. As in DSR, the initiator sets the id to an identifier that it has not recently used in initiating a Route Discovery. The time interval is the TESLA time interval at the pessimistic expected arrival time of the request at the target, accounting for clock skew. The initiator of the request then initializes the hash chain to $MAC_{KS,D}$ (initiator, target, id, time interval) and the node list and MAC list to empty lists.

When any node *A* receives a RREQ for which it is not the target, the node checks its local table of <initiator, id> values from recent requests it has received, to determine if it has already seen a request from this same Route Discovery. If it has, the node discards the packet, as in DSR. The node also checks whether the time interval in the request is valid: that time interval must not be too far in the future, and the key corresponding to it must not have been disclosed yet. If the time interval is not valid, the node discards the packet. Otherwise, the node modifies the request by appending its own address (*A*) to the node list in the request, replacing the hash chain field with H [A, *hash chain*], and appending a MAC of the entire REQUEST to the MAC list. The node uses the TESLA key K_{Ai} to compute the MAC, where *i* is the index for the time interval specified in the request. Finally, the node rebroadcasts the modified RREQ, as in DSR.

When the target node receives the RREQ, it checks the validity of the request by determining that the keys from the time interval specified have not been disclosed yet, and that the hash chain field is equal to:

$$\mathbf{H\,[\eta_n\,,\,H\,[\eta_{n-1}\,,\,H\,[\,\ldots\,,\,H\,[\eta_1\,,\,MAC_{KSD}\,(initiator,\,target,\,id,\,time\ interval)\,]\,\ldots\,]\,]\,]}$$

where η_i is the node address at position i of the node list in the request, and where n is the number of nodes in the node list. If the target node determines that the request is valid, it returns a RREP to the initiator, containing eight fields:

<ROUTE REPLY, target, initiator, time interval, node list, MAC list, target MAC, key list>

The target, initiator, time interval, node list, and MAC list fields are set to the corresponding values from the RREQ, the target MAC is set to a MAC computed on the preceding fields in the reply with the key KDS , and the key list is initialized to the empty list. The RREP is then returned to the initiator of the request along the source route obtained by reversing the sequence of hops in the node list of the request.

A node forwarding a RREP waits until it is able to disclose its key from the time interval specified, then it appends its key from that time interval to the key list field in the reply and forwards the packet according to the source route indicated in the packet. Waiting delays the return of the RREP but does not consume extra computational power.

When the initiator receives a RREP, it verifies that each key in the key list is valid, that the target MAC is valid, and that each MAC in the MAC list is valid. If all of these tests succeed, the node accepts the RREP; otherwise, it discards it.

In order to prevent the injection of invalid route errors into the network fabricated by any node other than the one on the sending end of the link specified in the error message, each node that encounters a broken link adds TESLA authentication information to the route error message, such that all nodes on the return path can authenticate the error. However, TESLA authentication is delayed, so all the nodes on the return path buffer the error but do not consider it until it is authenticated. Later, the node that encountered the broken link discloses the key and sends it over the return path, which enables nodes on that path to authenticate the buffered error messages.

ARIADNE is protected also from a flood of RREQ packets that could lead to the cache poisoning attack. Benign nodes can filter out forged or excessive RREQ packets using *Route Discovery chains*, a mechanism for authenticating route discovery, allowing each node to rate-limit discoveries initiated by any other node. The authors present two different approaches that can be found in [2].

ARIADNE is immune to the wormhole attack only in its advanced version: using the TIK (TESLA with Instant Key disclosure, references) protocol that allows for very precise time synchronization between the nodes of the network, it is possible to detect anomalies in routing traffic flows in the network.

2.2.3 ARAN

The ARAN secure routing protocol proposed by Dahill, Levine, Royer and Shields is conceived as an on-demand routing protocol that detects and protects against malicious actions carried out by third parties and peers in the ad hoc environment. ARAN introduces *authentication*, message *integrity* and *non-repudiation* as part of a minimal security policy for the ad hoc environment and consists of a preliminary certification process, a mandatory end-to-end authentication stage and an optional second stage that provides secure shortest paths.

ARAN requires the use of a trusted certificate server (T): before entering in the ad hoc network, each node has to request a certificate signed by T. The certificate contains the IP address of the node, its public key, a timestamp of when the certificate was created and a time at which the certificate expires along with the signature by T. All nodes are supposed to maintain fresh certificates with the trusted server and must know T’s public key.

The goal of the first stage of the ARAN protocol is for the source to verify that the intended destination was reached. In this stage, the source trusts the destination to choose the return path. A source node, *A*, initiates the route discovery process to reach the destination *X* by broadcasting to its neighbors a route discovery packet called RDP:

[RDP; IP_X ; $cert_A$; N_A ; t]K_{A-}

The RDP includes a packet type identifier (“RDP”), the IP address of the destination (IP_X), *A*'s certificate ($cert_A$), a nonce N_A , and the current time t, all signed with A's private key. Each time A performs route discovery, it monotonically increases the nonce.

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

[[RDP; IP_X ; $cert_A$; N_A ; t]K_{A-}]K_{B-} ; $cert_B$

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

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

[[RDP; IP_X ; $cert_A$; N_A ; t]K_{A-}]K_{C-} ; $cert_C$

Eventually, the message is received by the destination, *X*, who replies to the first RDP that it receives for a source and a given nonce. There is no guarantee that the first RDP received traveled along the shortest path from the source. The destination unicasts a Reply (REP) packet back along the reverse path to the source. Let the first node that receives the RDP sent by X be node *D*. *X* will send to *D* the following message:

[REP; IP_A ; $cert_X$; N_A ; t]K_{X-}

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

[[REP; IP_A ; $cert_X$; N_A ; t]K_{X-}]K_{D-} ; $cert_D$

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

[[REP; IP_A ; $cert_X$; N_A ; t]K_{X-}]K_{C-} ; $cert_C$

A node checks the signature of the previous hop as the REP is returned to the source. This avoids attacks where malicious nodes instantiate routes by impersonation and re-play of X's message. When the source receives the REP, it verifies that the correct nonce was returned by the destination as well as the destination's signature. Only the destination can answer an RDP packet. Other nodes that already have paths to the destination cannot reply for the destination. While other protocols allow this networking optimization, ARAN removes several possible exploits and cuts down on the reply traffic received by the source by disabling this option.

The second stage of the ARAN protocol guarantees in a secure way that the path received by a source initiating a route discovery process is the shortest. Similarly to the first stage of the protocol, the source broadcasts a *Shortest Path Confirmation* (SPC) message to its neighbors: the SPC message is different from the RDP message only in two additional fields that provide the destination *X* certificate and the encryption of the entire message with *X*'s public key (which is a costly operation). The onion-like signing of messages combined with the encryption of the data prevents nodes in the middle from changing the path length because doing so would break the integrity of SPC the packet.

Also the route maintenance phase of the ARAN protocol is secured by digitally signing the route error packets. However it is extremely difficult to detect when error messages are *fabricated* for links that are truly active and not broken. Nevertheless, because messages are signed, malicious nodes cannot generate error messages for other nodes. The non-repudiation provided by the signed error message allows a node to be verified as the source of each error message that it sends.

As with any secure system based on cryptographic certificates, the key revocation issue has to be addressed in order to make sure that expired or revoked certificates do not allow the holder to access the network. In ARAN, when a certificate needs to be revoked, the trusted certificate server T sends a broadcast message to the ad hoc group that announces the revocation. Any node receiving this message re-broadcast it to its neighbors. Revocation notices need to be stored until the revoked certificate would have expired normally. Any neighbor of the node with the revoked certificate needs to reform routing as necessary to avoid transmission through the now un-trusted node. This method is not failsafe. In some cases, the un-trusted node that is having its certificate revoked may be the sole connection between two parts of the ad hoc network. In this case, the un-trusted node may not forward the notice of revocation for its certificate, resulting in a partition of the network, as nodes that have received the revocation notice will no longer forward messages through the un-trusted node, while all other nodes depend on it to reach the rest of the network. This only lasts as long as the un-trusted node's certificate would have otherwise been valid, or until the un-trusted node is no longer the sole connection between the two partitions. At the time that the revoked certificate should have expired, the un-trusted node is unable to renew the certificate, and routing across that node ceases. Additionally, to detect this situation and to hasten the propagation of revocation notices, when a node meets a new neighbor, it can exchange a summary of its revocation notices with that neighbor; if these summaries do not match, the actual signed notices can be forwarded and re-broadcasted to restart propagation of the notice.

The ARAN protocol protects against exploits using *modification, fabrication* and *impersonation* but the use of asymmetric cryptography makes it a very costly protocol to

use in terms of CPU and energy usage. Furthermore, ARAN is not immune to the *wormhole* attack

2.2.4 SEAD

Hu, Perrig and Johnson present a *proactive* secure routing protocol based on the Destination-Sequenced Distance Vector protocol (DSDV). In a proactive (or periodic) routing protocol nodes periodically exchange routing information with other nodes in attempt to have each node always know a current route to all destinations [7]. Specifically, SEAD is inspired by the DSDV-SQ version of the DSDV protocol. The DSDV-SQ version of the DSDV protocol has been shown to outperform other DSDV versions in previous ad hoc networks simulations [8, 9].

SEAD deals with attackers that *modify* routing information broadcasted during the update phase of the DSDV-SQ protocol: in particular, routing can be disrupted if the attacker modifies the sequence number and the metric field of a routing table update message. *Replay attacks* are also taken into account.

In order to secure the DSDV-SQ routing protocol, SEAD makes use of efficient *one-way hash chains* rather than relaying on expensive asymmetric cryptography operations. However, like the other secure protocols presented in this chapter, SEAD assumes some mechanism for a node to distribute an authentic element of the hash chain that can be used to authenticate all the other elements of the chain. As a traditional approach, the authors suggest to ensure the key distribution relaying on a trusted entity that signs public key certificates for each node; each node can then use its public key to sign a hash chain element and distribute it.

The basic idea of SEAD is to authenticate the sequence number and metric of a routing table update message using hash chains elements. In addition, the receiver of SEAD routing information also authenticates the sender, ensuring that the routing information originates form the correct node.

To create a one-way hash chain, a node chooses a random initial value $x \in \{0,1\}^{\rho}$, where ρ is the length in bits of the output of the hash function, and computes the list of values $h_0,h_1,h_2,h_3,\ldots,h_n$, where $h_0=x$, and $h_i = H(h_{i-1})$ for $0< i \leq n$, for some n. As an example, given an authenticated h_i value, a node can authenticate h_{i-3} by computing $H(H(H(h_{i-3})))$ and verifying that the resulting value equals hi.

Each node uses a specific authentic (i.e. signed) element from its hash chain in each routing update that it sends about itself (metric 0). Based on this initial element, the one-way hash chain provides authentication for the lower bound on the metric in other routing updates for that node. The use of a hash value corresponding to the sequence number and metric in a routing update entry prevents any node from advertising a route to some destination claiming a greater sequence number than that destination's own current sequence number. Likewise, a node can not advertise a route better than those for which it has received an advertisement, since the metric in an existing route cannot be decreased due to the on-way nature of the hash chain.

When a node receives a routing update, it checks the authenticity of the information for each entry in the update using the destination address, the sequence number and the metric of the received entry, together with the latest prior *authentic* hash

value received from that destination's hash chain. Hashing the received elements the correct number of times (according to the prior authentic hash value) assures the authenticity of the received information if the calculated hash value and the authentic hash value match.

The source of each routing update message in SEAD must also be authenticated, since otherwise, an attacker may be able to create routing loops through the *impersonation* attack. The authors propose two different approaches to provide node authentication: the first is based on a broadcast authentication mechanism such as TESLA, the second is based on the use of Message Authentication Codes, assuming a shared secret key between each couple of nodes in the network.

SEAD does not cope with *wormhole* attacks though the authors propose, as in the ARIADNE protocol, to use the TIK protocol to detect the threat.

2.3 Notes on the wormhole attack

The wormhole attack is a severe threat against ad hoc routing protocols that is particularly challenging to detect and prevent. In a wormhole attack a malicious node can record packets (or bits) at one location in the network and tunnel them to another location through a private network shared with a colluding malicious node. Most existing ad hoc routing protocols, without some mechanism to defend them against the wormhole attack, would be unable to find consistent routes to any destination, severely disrupting communication.

A dangerous threat can be perpetrated if a wormhole attacker tunnels all packets through the wormhole honestly and reliably since no harm seems to be done: the attacker actually seems to provide a useful service in connecting the network more efficiently. However, when an attacker forwards only routing control messages and not data packets, communication may be severely damaged. As an example, when used against an on demand routing protocol such as DSR, a powerful application of the wormhole attack can be mounted by tunneling each RREQ message directly to the destination target node of the request. This attack prevents routes more than two hops long from being discovered because RREP messages would arrive to the source faster than any other replies or, worse, RREQ messages arriving from other nodes next to the destination than the attacker would be discarded since already seen.

Hu, Perrig and Johnson propose an approach to detect a wormhole based on *packet leashes* [10]. The key intuition is that by authenticating either an extremely precise timestamp or location information combined with a loose timestamp, a receiver can determine if the packet has traversed a distance that is unrealistic for the specific network technology used.

Temporal leashes rely on extremely precise time synchronization and extremely precise timestamps in each packet. The travel time of a packet can be approximated as the difference between the receive time and the timestamp. Given the precise time synchronization required by temporal leashes, the authors propose efficient broadcast authenticators based on symmetric primitives. In particular they extend the TESLA broadcast authentication protocol to allow the disclosure of the authentication key within the packet that is authenticated.

Geographical leashes are based on location information and loosely synchronized clocks. If the clocks of the sender and the receiver are synchronized within a certain threshold and the velocity of any node is bounded, the receiver can compute an upper bound on the distance between the sender and itself and use it to detect anomalies in the traffic flow. In certain circumstances however, bounding the distance between the sender and the receiver cannot prevent wormhole attacks: when obstacles prevent communication between two nodes that would otherwise be in transmission range, a distance-based scheme would still allow wormholes between the sender and the receiver. To overcome this problem, in a variation of the geographical leashes the receiver verifies that every possible location of the sender can reach every possible location of the receiver based on a radio propagation model implemented in every node.

In some special cases, wormholes can also be detected through techniques that don't require precise time synchronization nor location information. As an example, it would be sufficient to modify the routing protocol used to discover the path to a destination so that it could handle multiple routes: a verification mechanism would then detect anomalies when comparing the metric (e.g. number of hops) associated to each route. Any node advertising a path to a destination with a metric considerably lower than all the others could raise the suspect of a wormhole.

Furthermore, if the wormhole attack is performed only on routing information while dropping data packets, other mechanisms can be used to detect this misbehavior. When a node doesn't correctly participate to the network operation by not executing a particular function (e.g. packet forwarding) a collaborative monitoring technique can detect and gradually isolate misbehaving nodes. Lack of cooperation and security mechanism used to enforce node cooperation to the network operation is the subject of the next section.

3. Cooperation enforcement in Mobile Ad Hoc Networks

As opposed to networks using dedicated nodes to support basic networking functions like packet forwarding and routing, in ad hoc networks these functions are carried out by all available nodes in the network. There is no reason, however, to assume that the nodes in the network will eventually cooperate one with another since network operation consumes energy, a particularly scarce resource in a battery powered environment like MANET. The new type of node misbehavior that is specific to ad hoc networks is caused by lack of cooperation and goes under the name of *node selfishness*. A selfish node does not directly intend to damage other nodes with active attacks (mainly because performing active attacks can be very expensive in terms of energy consumption) but it simply does not cooperate to the network operation, saving battery life for its own communications.

Damages provoked by selfish behavior can not be underestimated: a simulation study present in the literature [11] shows the impact of a selfish behavior in terms of global network throughput and global communication delay when the DSR routing protocol is used. The simulation results show that even a little percentage of selfish nodes present in the network leads to a severe degradation of performances. Furthermore, any

security mechanism that tries to enforce cooperation among the nodes should focus not only on one particular function, but on both the routing and the packet forwarding function: as an example, if a source routing mechanism as DSR is used, any node that does not participate to the routing protocol cannot claim to participate to the packet forwarding function since it cannot appear in any route, meaning that it will never be asked to relay packets for other nodes of the network.

The node selfishness problem has only recently been addressed by the research community, and still few mechanisms are provided to combat such misbehavior. Mechanisms that enforce node cooperation in a MANET can be divided in two categories: the first is currency based (see section 3.1) and the second uses a local monitoring technique (see sections 3.2, 3.3). On one side, currency based systems are simple to implement but rely on a tamperproof hardware. The main drawback of this approach resides in the difficulty to establish how the virtual currency has to be exchanged making their use not realistic in a practical system. On the other side, cooperative security schemes based on local monitoring seem to offer the most suitable solution to the selfishness problem. Every node of the MANET monitors its local neighbors evaluating for each of them a metric that is directly related to the nodes' behavior. Based on that metric, a selfish node can be gradually isolated from the network. The main drawback of this second approach is related to the absence of a mechanism that securely identifies the nodes of the network: any selfish node could elude the cooperation enforcement mechanism and get rid of its bad reputation just by changing its identity.

Following, the main research efforts towards the solution of the node selfishness problem are presented.

3.1 Nuglets

In [14],] Buttyan and Hubaux present two important issues targeted specifically at the ad hoc networking environment: first, end-users must be given some incentive to cooperate to the network operation (especially to relay packets belonging to other nodes); second, end-users must be discouraged from overloading the network. The solution presented in their paper consists in the introduction of a virtual currency (that they call Nuglets) used in every transaction. Two different models are described: the Packet Purse Model and the Packet Trade Model. In the Packet Purse Model each packet is loaded with nuglets by the source and each forwarding host takes out nuglets for its forwarding service. The advantage of this approach is that it discourages users from flooding the network but the drawback is that the source needs to know exactly how many nuglets it has to include in the packet it sends. In the Packet Trade Model each packet is traded for nuglets by the intermediate nodes: each intermediate node buys the packet from the previous node on the path. Thus, the destination has to pay for the packet. The direct advantage of this approach is that the source does not need to know how many nuglets need to be loaded into the packet. On the other hand, since the packet generation is not charged, malicious flooding of the network cannot be prevented. There are some further issues that have to be solved: concerning the Packet Purse Model, the intermediate nodes are able to take out more nuglets than they are supposed to; concerning the Packet Trade

Model, the intermediate nodes are able to deny the forwarding service after taking out nuglets from a packet.

3.2 Confidant

The acronym given to the cooperation mechanism proposed by Buchegger and Le Boudec stands for “Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks” [15,16] and it detects malicious nodes by means of observation or reports about several types of attacks, thus allowing nodes to route around misbehaved nodes and to isolate them. CONFIDANT works as an extension to a routing protocol such as Dynamic Source Routing (DSR).

Nodes have a monitor for observations, reputation records for first-hand and trusted second-hand observations about routing and forwarding behavior of other nodes, trust records to control trust given to received warnings, and a path manager to adapt their behavior according to reputation and to take action against malicious nodes. The term reputation is used to evaluate routing and forwarding behavior according to the network protocol, whereas the term trust is used to evaluate participation in the CONFIDANT meta-protocol.

The dynamic behavior of CONFIDANT is as follows. Nodes monitor their neighbors and change the reputation accordingly. If they have reason to believe that a node misbehaves, they can take action in terms of their own routing and forwarding and they can decide to inform other nodes by sending an ALARM message. When a node receives such an ALARM either directly or by promiscuously listening to the network, it evaluates how trustworthy the ALARM is based on the source of the ALARM and the accumulated ALARM messages about the node in question. It can then decide whether to take action against the misbehaved node in the form of excluding routes containing the misbehaved node, re-ranking paths in the path cache, reciprocating by non-cooperation, and forwarding an ALARM about the node.

The first version of CONFIDANT was, despite the filtering of ALARM messages in the trust manager, vulnerable to concerted efforts of spreading wrong accusations. This problem has been addressed by the use of Bayesian statistics for classification and the exclusion of liars.

Simulations with nodes that do not participate in the forwarding function have shown that CONFIDANT can cope well, even if half of the network population acts maliciously. Further simulations concerning the effect of second-hand information and slander have shown that slander can effectively be prevented while still retaining a significant detection speed-up over using merely first-hand information.

The limitations of CONFIDANT lie in the assumptions for detection-based reputation systems. Events have to be observable and classifiable for detection, and reputation can only be meaningful if the identity of each node is persistent, otherwise it is vulnerable to spoofing attacks.

3.3 CORE

The security scheme proposed by Michiardi and Molva [18, 19], stimulates node cooperation by a collaborative monitoring technique and a reputation mechanism. Each node of the network monitors the behavior of its neighbors with respect to a requested function and collects observations about the execution of that function: as an example, when a node initiates a Route Request (e.g., using the DSR routing protocol) it monitors that its neighbors process the request, whether with a Route Reply or by relaying the Route Request. If the observed result and the expected result coincide, then the observation will take a positive value, otherwise it will take a negative value.

Based on the collected observations, each node computes a reputation value for every neighbor using a sophisticated reputation mechanism that differentiates between subjective reputation (observations), indirect reputation (positive reports by others), and functional reputation (task-specific behavior), which are weighted for a combined reputation value. The formula used to evaluate the reputation value avoids false detections (caused for example by link breaks) by using an aging factor that gives more relevance to past observations: frequent variations on a node behavior are filtered. Furthermore, if the function that is being monitored provides an acknowledgement message (e.g., the Route Reply message of the DSR protocol), reputation information can also be gathered about nodes that are not within the radio range of the monitoring node. In this case, only positive ratings are assigned to the nodes that participated to the execution of the function in its totality.

The CORE mechanism resists to attacks performed using the security mechanism itself: no negative ratings are spread between the nodes, so that it is impossible for a node to maliciously decrease another node's reputation. The reputation mechanism allows the nodes of the MANET to gradually isolate selfish nodes: when the reputation assigned to a neighboring node decreases below a pre-defined threshold, service provision to the misbehaving node will be interrupted. Misbehaving nodes can, however, be reintegrated in the network if they increase their reputation by cooperating to the network operation.

As for the other security mechanism based on reputation the CORE mechanism suffers from the spoofing attack: misbehaving nodes are not prevented from changing their network identity allowing the attacker to elude the reputation system. Furthermore, no simulation results prove the robustness of the protocol even if the authors propose an original approach based on game theory in order to come up with a formal assessment of the security properties of CORE.

3.4 Token-based cooperation enforcement

In the approach presented by Yang, Meng, Lu [20] each node of the ad hoc network has a token in order to participate in the network operations, and its local neighbors collaboratively monitor it to detect any misbehavior in routing or packet forwarding services. Upon expiration of the token, each node renews its token via its multiple neighbors: the period of validity of a node's token is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its token less and less frequently as time evolves.

The security solution proposed by the authors is composed of four closely interacted components: the *neighbor verification*, which describes how to verify whether each node in the network is a legitimate or malicious node, the *neighbor monitoring*, which describes how to monitor the behavior of each node in the network and detect occasional attacks from malicious nodes, the *intrusion reaction*, which describes how to alert the network and isolate the attackers and the *security enhanced routing protocol*, which explicitly incorporates the security information collected by the other components into the ad hoc routing protocol.

Concerning the token issuing/renewal phase, the authors assume a global secret/public key pair SK/PK, where PK is well known by every node of the network. SK is shared by *k* neighbors who collaboratively sign the token requested or renewed by local nodes. Token verification, on the other side, follows three steps: 1) identity match between the node's ID and the token ID, 2) validity time verification, 3) issuer signature. If the token verification phase fails, the corresponding node is rejected from the network and both routing and data packets are dropped for that node.

Routing security relies on the redundancy of routing information rather than cryptographic techniques: the routing protocol that the authors use as a basis is the Ad hoc On demand Distance Vector protocol (AODV) which is extended in order to detect false routing update messages by comparing routing information gathered from different neighboring nodes. Packet forwarding misbehavior is also detected using a modified version of the watchdog technique presented in [17] bypassing the absence of any source route information by adding a next hop field in the routing messages.

The proposed solution presents some drawbacks: the bootstrap phase needed to generate a valid collection of partial tokens which will be used by a node to create its final token has some limitations. For example to the number of neighbors necessary to complete the signature of every partial token has to be at least *k*, suggesting the use of such security mechanism in rather large and dense ad hoc networks. On the other side, the validity period of a token increases proportionally to the time during which the node behave well: this interesting feature has less impact if node mobility is high. Frequent changes in the local subset of the network that shares a key for issuing valid tokens can cause high computational overhead, not to mention the high traffic generated by issuing/renewing a token, suggesting that the token-based mechanism is more suitable in ad hoc networks where node mobility is low. Spoofing attacks, where a node can request more than one token claiming different identities, are not taken into account even if the authors suggest that MAC addresses can be sufficient for node authentication purposes.

4. Authentication and Public Key Infrastructure (PKI)

Providing security support for ad hoc networks is challenging for a number of reasons: wireless networks are susceptible to security attacks ranging from passive eavesdropping to active interfering and DoS attacks; occasional break-ins in a large-scale mobile network are inevitable over a large time interval; ad hoc networks provide no infrastructure support; mobile nodes may constantly leave or join the network; mobility-induced wireless links breakage/reconnection and wireless channel errors make timely

communications over multiple hops highly unreliable; and a scalable solution is a must for a large scale network. However, the provision of basic security services such as authentication, confidentiality, integrity and non-repudiation is critical in order to deploy the mobile wireless ad hoc technology in commercial and military environments.

Authentication services specific to the ad hoc environment have been recently studied by the research community in order to come up with a fully self-organized architecture to overcome the limitations intrinsic to the secure routing protocols that have been presented in section 2.2.

The basic assumption adopted by some secure routing protocol such as SRP is the existence of an a-priori security association between all the communicating nodes of the network: the limitations introduced by this approach range from the need of a managed environment, such as a common authority that pre-charges all the mobile terminals with a secret key shared by every couple of communicating nodes, to scalability problems.

Other secure routing protocols (such as Ariadne) rely on an initialization phase during which a well known trusted third party (TTP) issues public key certificates used to authenticate (together with the private key of each certificate holder) hash chain elements that will be subsequently used to provide some low cost (in terms of CPU usage) authentication services. In this case, the use of such a secure protocol is not limited to the managed environment, and the open environment can be targeted: indeed, it is not necessary for the mobile nodes that form the ad hoc network to be managed by the same authority that provides the initial authentication setup. However, the bootstrap phase requires an external infrastructure, which has to be available also during the lifetime of the ad hoc network to provide revocation services for certificates that have expired or that have been explicitly revoked.

Current efforts in order to provide scalable, fully self-organized public key infrastructure and authentication services can be classified in two categories, one based on a PGP-like architecture and one based on the polynomial secret sharing technique, and are presented in the next sections.

4.1 Self-Organized Public-Key Management based on PGP

Capkun, Buttyan and Hubaux propose a fully self-organized public key management system that can be used to support security of ad hoc network routing protocols [21]. The suggested approach is similar to PGP [22] in the sense that users issue certificates for each other based on their personal acquaintances. However, in the proposed system, certificates are stored and distributed by the users themselves, unlike in PGP, where this task is performed by on-line servers (called certificate directories). In the proposed self-organizing public-key management system, each user maintains a *local certificate repository*. When two users want to verify the public keys of each other, they merge their local certificate repositories and try to find appropriate certificate chains within the merged repository that make the verification possible.

The success of this approach very much depends on the construction of the local certificate repositories and on the characteristics of the certificate graphs. By a certificate graph is meant to be a graph whose vertices represent public-keys of the users and the

edges represent public-key certificates issued by the users. The authors investigate on several repository construction algorithms and study their performance. The proposed algorithms take into account the characteristics of the certificate graphs in a sense that the choice of the certificates that are stored by each mobile node depends on the connectivity of the node and its certificate graph neighbors.

More precisely, each node stores in its local repository several directed and mutually disjoint paths of certificates. Each path begins at the node itself, and the certificates are added to the path such that a new certificate is chosen among the certificates connected to the last node on the path (initially the node that stores the certificates), such that the new certificate leads to the node that has the highest number of certificates connected to it (i.e., the highest vertex degree). The authors call this algorithm the *Maximum Degree Algorithm*, as the local repository construction criterion is the degree of the vertices in a certificate graph.

In a second, more sophisticated algorithm that is called the *Shortcut Hunter Algorithm*, certificates are stored into the local repositories based on the number of the shortcut certificates connected to the users. The shortcut certificate is a certificate that, when removed from the graph makes the shortest path between two users previously connected by this certificate strictly larger than two.

When verifying a certificate chain, the node must trust the issuer of the certificates in the chain for correctly checking that the public key in the certificate indeed belongs to the node identification (ID) named in the certificate. When certificates are issued by the mobile nodes of an ad hoc network instead of trusted authorities, this assumption becomes unrealistic. In addition, there may be malicious nodes who issue false certificates. In order to alleviate these problems, the authors propose the use of authentication metrics [23]: it is not enough to verify a node ID key binding via a single chain of certificates. The authentication metric is a function that accepts two keys (the verifier and the verified node) and a certificate graph and returns a numeric value corresponding to the degree of authenticity of the key that has to be verified: one example of authentication metric is the number of disjoint chains of certificates between two nodes in a certificate graph.

The authors emphasize that before being able to perform key authentication, each node must first build its local certificate repository, which is a relatively expensive operation (in terms of bandwidth and time). However this initialization phase must be performed rarely and once the certificate repositories have been built, then any node can perform key authentication using only local information and the information provided by the targeted node. It should also be noted that local repositories become obsolete if a large number of certificate are revoked, as then the certificate chains are no longer valid; the same comment applies in the case when the certificate graph changes significantly. Furthermore, PGP-like schemes are more suitable for small communities because that the authenticity of a key can be assured with a higher degree of trustiness. The authors propose the use of authentication metrics to alleviate this problem: this approach however provides only probabilistic guarantees and is dependent on the characteristics of the certificate graph on which it operates. The authors also carried out a simulation study showing that for the certificate graphs that are likely to emerge in self-organized systems, the proposed approach yields good performances both in terms of the size of the local repository stored in each node and scalability.

4.2 Ubiquitous and Robust Authentication Services based on polynomial secret sharing

In [24] Luo and Lu present a mechanism that provides ubiquitous authentication service availability by taking a *certificate-based* approach. In the proposed scheme, any two communicating nodes can establish a temporary trust relationship via globally verifiable certificates. With a *scalable threshold sharing* of the certificate signing key, certification services (issuing, renewal and revocation) are distributed among each node in the network: a single node holds just a share of the complete certificate signing key. While no single node has the power of providing full certification services, multiple nodes in a network locality can collaboratively provide such services.

The authors propose a *localized trust model* to characterize the localized nature of security concerns in large ad hoc wireless networks. When applying such trust model, an entity is trusted if any *k* trusted entities claim so: these *k* trusted entities are typically the neighboring nodes of the entity. A locally trusted entity is globally accepted and a locally distrusted entity is regarded untrustworthy anywhere. *k* is a system wide parameter that sets the global acceptance criteria and should be honored by each entity in the system.

The basic assumption that are necessary for the security mechanism to function properly are: 1) each node has a unique nonzero identifier (ID), such as its MAC layer address; 2) each node has some one-hop discovery mechanism; 3) each node has at least k one-hop legitimate neighboring nodes, or the network has a minimum density of well-behaving nodes; 4) each node is equipped with some detection mechanism to identify misbehaving nodes among its one-hop neighborhood; 5) the mobility is characterized by a maximum node moving speed S_{max}.

In the security architecture proposed [24], each node carries a certificate signed by the share certificate signing key SK, while the corresponding public key PK is assumed to be well-known by all the nodes of the network, so that certificates are globally verifiable. Nodes without valid certificates will be isolated, that is, their packets will not be forwarded by the network. Essentially, any node without a valid certificate are treated the same as adversaries. When a mobile node moves to a new location, it exchanges certificates with its new neighbors and goes through mutual authentication process to build trust relationships. Neighboring nodes with such trust relationship help each other to forward and route packets. They also monitor each other to detect possible attacks and break-ins. Specific monitoring algorithms and mechanisms are left to each individual node's choice. When a node requests a signed certificate by the coalition of *k* nodes, each of the certificate issuing nodes checks its record on the requesting node. If the record shows that the requestor is a well-behaving legitimate node it returns a partial certificate by applying its share of SK. Otherwise the request is dropped. By collecting *k* partial certificates the requesting node combines them together to generate the full new certificate as if it was issued from a certification authority server. A misbehaving or broken node that is detected by its neighbors will be unable to get a new certificate.

The security of the certificate signing key SK is protected by a k-threshold polynomial sharing mechanism. However, this technique requires a bootstrapping phase where a "dealer" has to privately send to each node its share of the SK. The authors propose a scalable initialization mechanism that they called "self-initialization". In this case, the dealer is only responsible to initialize the very first *k* nodes, no matter how large

the network would be. The initialized nodes collaboratively initialize other nodes: repeating this procedure, the network progressively self-initializes itself. The same mechanism is applied when new nodes join the network.

Certificate revocation is also handled by the proposed architecture and an original approach to handle roaming adversaries is presented: without this additional mechanism, any misbehaving node that moves to a location of the network where its new neighbors have no information in their monitoring records about the attacker, could get a new valid certificate. Roaming nodes are defeated with the flooding of “accusation” messages that travel in the network and inform distant nodes about the behavior of a suspect node. Accusation messages are accepted only if they come from well-behaving nodes and have a specific time to live (TTL) field that is calculated based on the maximum node speed specified in the assumptions.

The main drawbacks of the proposed architecture range from the necessity of an external, trusted dealer that initializes the very first *k* nodes of a coalition to the choice of the system-wide parameter *k*. To cope with the first problem, the authors propose to use a distributed RSA key pair generation [25] for the very first *k* nodes. On the other side, no practical solutions are presented to cope with the strong assumption that every node of the network has at least *k* trusted and not compromised neighbors. This limitation makes the proposed architecture un-useful for all the nodes that are located at the perimeter of the ad hoc network. More over, the authors assume that any new node that joins the system already has an initial certificate: initial certificates can be obtained in two ways. Every node may be issued an initial certificate by an offline authority or every new node may use any coalition of *k* neighbors to issue the initial certificate via a collaborative admission control mechanism. These problems reduce the effectiveness of the proposed architecture as a fully self organized infrastructure.

5. Security Mechanisms in Layer 2

This section presents data link layer security solutions that are suitable for MANET. The most prevalent solutions are the security mechanisms as part of 802.11 and Bluetooth specifications. Further to a detailed overview of their specifications the weaknesses of each solution is analyzed. The relevance of these solutions with respect to the security requirements of MANET are then discussed.

5.1 Wired Equivalent Privacy (WEP)

The first security scheme provided in the series of IEEE 802.11 standards is Wired Equivalent Privacy (WEP) specified as part of the 802.11b Wireless Fidelity (Wi-Fi) standard [26]. WEP was originally designed to provide security for wireless local area networks (WLAN) with a level of protection that is similar to the one expected in wired LAN’s. The latter enjoys security and privacy due to the physical security mechanisms like building access control. Physical security mechanisms unfortunately do not prevent eavesdropping and unauthorized access in case of wireless communications. WEP thus aims at covering the lack of physical security akin to WLANs with security mechanisms

based on cryptography. Unfortunately WEP suffers from various design flaws and some exposure in the underlying cryptographic techniques that seriously undermine its security claims.

5.1.1 WEP Security Mechanisms

WEP security mechanisms include data encryption and integrity. Both mechanisms are handled simultaneously for each frame as illustrated in

F

igure 1.

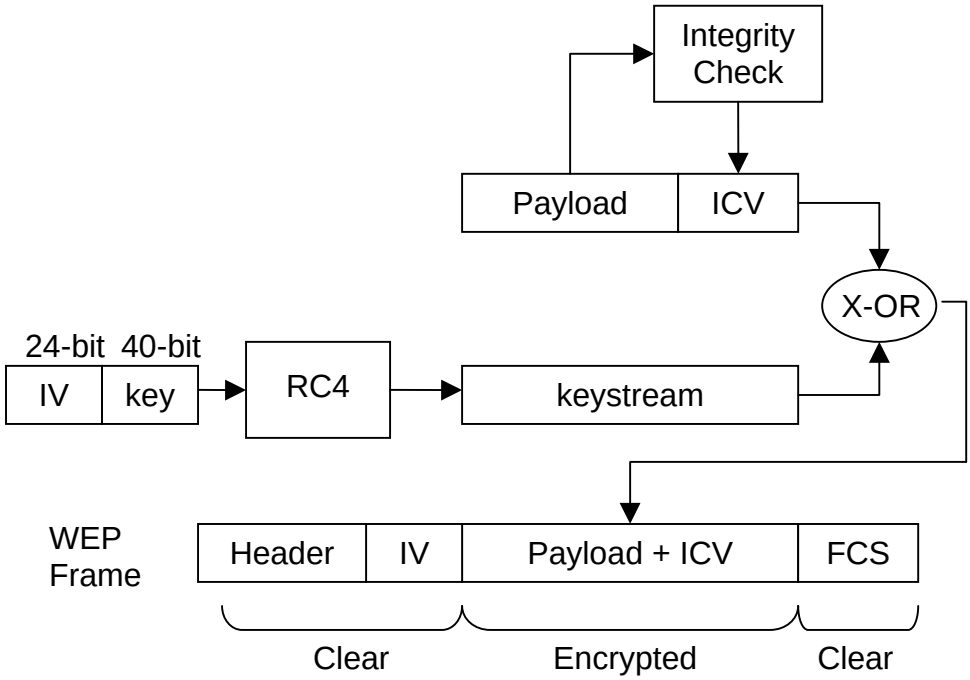

Figure 1: WEP Frame Security Mechanisms

To prepare a protected frame, first an integrity check value (ICV) of the frame payload is computed using a cyclic redundancy check (CRC) function. The cleartext payload concatenated with ICV is then encrypted using a bit-wise exclusive-or operation with a keystream as long as the payload concatenated with ICV. The keystream is a pseudo-random bit stream generated by the RC4 [28] algorithm from a 40-bit secret key prepended with a 24-bit Initialization Value (IV). The resulting protected frame includes the cleartext frame header, the cleartext IV, the result of the encryption and a cleartext frame check sequence field.
The recipient of a WEP frame first generates the keystream with RC4 using the shared secret key and the IV value retrieved from the received frame. The resulting keystream is exclusive-ored with the encrypted field of the frame to decrypt the payload and the ICV. The integrity of the payload is then checked by comparing the integrity check computed on the cleartext payload with the ICV resulting from the decryption.
The secret key can either be a default key shared by all the devices of a WLAN or a pair-wise secret shared only by two communicating devices. Since WEP does not provide any support for the exchange of pair-wise secret keys, the secret key must be manually installed on each device.

5.1.1.1 Security Problems in WEP

WEP suffers from many design flaws and some weaknesses in the way the RC4 cipher is used in WEP.
Data encryption in WEP is based on an approximation of the “one-time pad” [28] algorithm that can guarantee perfect secrecy under some circumstances. Like WEP

encryption, one-time pad encryption consists of the bit-wise exclusive-or between a binary plaintext message and a binary keystream as long as the message. The secrecy of the resulting ciphertext is perfect provided that each new message is encrypted with a different secret random keystream. The secrecy is not guaranteed when the keystream is re-used or its values can be predicted. Hence a first class of attacks on WEP exploit possible weaknesses in WEP's keystream generation process that make the secret keystream easily predictable or cause its re-use.
The first type of exposure is due to the likeliness of keystream re-use between a pair of communicating devices. Using the same secret key, the only variation in the input to the keystream generator is due to the variation in the IV. Since the IV is 24-bit value sent in cleartext, the re-use of a keystream can be easily detected. The re-use of a keystream is also very likely because of the small set of possible IV values that can be exhausted in a few hours in busy traffic between two nodes. This type of exposure gets even worse if some care is not taken during the implementation of the standard: some products set the IV to a constant value (0 or 1) at the initialization of the encryption process for each frame sequence. The second type of exposure is due to the use of a 40-bit secret that is highly vulnerable to exhaustive search with current computational power.
WEP data encryption is also exposed through an advanced attack that takes into account the characteristics of the RC4 algorithm [29] and drastically reduces the set of possible keystream values based on the attacker's ability to recover the first byte of encrypted WEP payload.
Another class of exposure on WEP concerns the data integrity mechanism using CRC in combination with the one-time pad encryption. Encryption using exclusive-or operation is transparent with respect to modification in that flipping bits of the encrypted message cause flipped bits on the same positions of the cleartext values resulting from decryption. As opposed to a cryptographically secure hash function, an integrity check computed with CRC yields predictable changes on the ICV with respect to single-bit modifications on the input message. Combining the transparency of exclusive-or with the predictable modification property of CRC, an attacker can flip bits on well known positions of an encrypted WEP payload and on the corresponding positions of the encrypted ICV so that the resulting cleartext payload is modified without the modification being detected by the recipient. It should be noted that the transparent modification of the WEP payload does not require the knowledge of the secret payload value since the attacker only needs to know the location of some selected fields in the payload to force the tampering of their value.
Last weakness of WEP is the lack of key management that is a potential exposure for most attacks exploiting manually distributed secrets shared by large populations.

5.1.1.2 New Proposal

To address the shortcomings of WEP IEEE has set up a special Task Group I (TGi) in charge of designing the new security architecture as part of the forthcoming version of the standard called 802.11i . To cope with brute force attacks, TGi has already proposed to include a 128-bit RC4 seed of which 104 bits are secret. TGi also proposed a long term architecture based on the IEEE 802.1x standard, which itself is based on the IETF's Extensible Authentication Protocol (EAP). IEEE 802.1x has a flexible design supporting

various authentication modes. However the new proposal based on 802.1x already suffers from problems like lack of data integrity for wireless frames and lack of mutual authentication.

5.2 Bluetooth Security Mechanisms

Bluetooth specification [27] includes a set of security profiles defined for the application layer in the so-called service level security and security profiles for the data link layer. Both types of profiles rely on key management, authentication and confidentiality services based on cryptographic security mechanisms implemented in the data link layer. Each Bluetooth device stands for an independent party from the point of view of the security protocols. In each device security mechanisms use a set of basic components:

- the device address (BD_ADDR): a-48 bit address defined by the IEEE that is unique for each Bluetooth device
- a 128-bit authentication key
- a128-bit symmetric data encryption key
- a random number (RAND) generated by a pseudo-random or (physical) random number generator.

5.2.1 Key Management

Bluetooth key management services provide each device with a set of symmetric cryptographic keys required for the initialization of a secret channel with another device, the execution of an authentication protocol, and the exchange of encrypted data with another device.

5.2.1.1 Key hierarchy

The key hierarchy of Bluetooth includes two generic key types:

- the *link key* that is shared by two or more parties and used as a key encrypting key (KEK) to encrypt other keys during key exchange or as a seed to generate other keys
- the *encryption key* that is a shared data encryption key (DEK).

Both the link key and the encryption key are 128-bit symmetric keys. The link key can further be qualified as an *initialization key*, a *unit key*, a *combination key* or a *master key*.

When two devices need to communicate using link level security and have no prior engagement, they establish a secure channel based on the *initialization key*. This channel is then used by the communicating devices to establish a *semi-permanent* link key that will be used several times to assure further key exchange between the devices. The *initialization key* is not used beyond the first key exchange. Each communicating device generates the *initialization key* using a pseudo-random number generator seeded with a secret personal identification number (PIN) entered by the user of each device and the value of RAND exchanged between the devices. In order for two communicating

devices to generate the same value for the initialization key, the same PIN value must therefore be entered on both devices.

Based on the key generation and storage capabilities of each communicating device the *semi-permanent* shared link key can either be a *unit key* or a *combination key* depending on the key generation and storage capabilities of communicating devices. The *unit key* is a device specific key that is generated during the initialization of each device. Its value changes rarely. It is generated using a pseudo-random number generator seeded with RAND and BD_ADDR (device address). The *combination key* is a pairwise key computed by two communicating devices based on a device specific key generated by each device. In order to compute a *combination key* first each device generates a device specific key based on RAND and BD_ADDR, the resulting keys are then exchanged between the pair of devices using the secure channel encrypted with the *initialization key*. The *combination key* is then derived by each of the pair of devices based on a simple combination of the two device specific keys.

The type of the link key to be used on a pairwise connection between two communicating devices is negotiated during link establishment. If one of the devices has restricted storage then this device's unit key is used as the pairwise link key, with obvious drawbacks due to the widespread disclosure of a semi-permanent device specific key. If both communicating devices have sufficient computing (key generation) and storage capabilities then they choose to use a combination key as a pairwise link key that has a different value for each pair of entities.

In a master-slave scenario, a short-lived link key called *master key* can be used between the master device and the slave devices. The lifetime of a *master key* is limited to the duration of the master-slave session. The *master key* is generated by the master device using pseudo-random number generator seeded by RAND and PIN. The resulting key is distributed through a channel secured under the *initialization key* to each slave the master wants to share the *master key* with.

The *encryption key* is generated by a pair of devices that share a link key using a pseudo-random number generator seeded by the link key, the random number RAND generated by one of the devices and transmitted to the other device prior to encrypted data exchange, and the secret Authenticated Ciphering Offset (ACO) generated by each device during the authentication process.

5.2.2 Authentication

The Bluetooth authentication scheme is based on a challenge-response protocol as depicted in Figure 2. When device A wants to authenticate device B using this protocol, A generates a random number (RAND) and sends it to B as a challenge then both devices compute a result (SRES) using the authentication algorithm E1 with RAND, the link key, and the device address of B. B then sends A SRES as the response to A's challenge RAND. B is successfully authenticated if the result SRES' computed by A matches SRES. During authentication each device also obtains the Authenticated Ciphering Offset (ACO) generated by the authentication algorithm E1. The ACO value is further used to generate the data encryption key that will be used between the pair of devices.

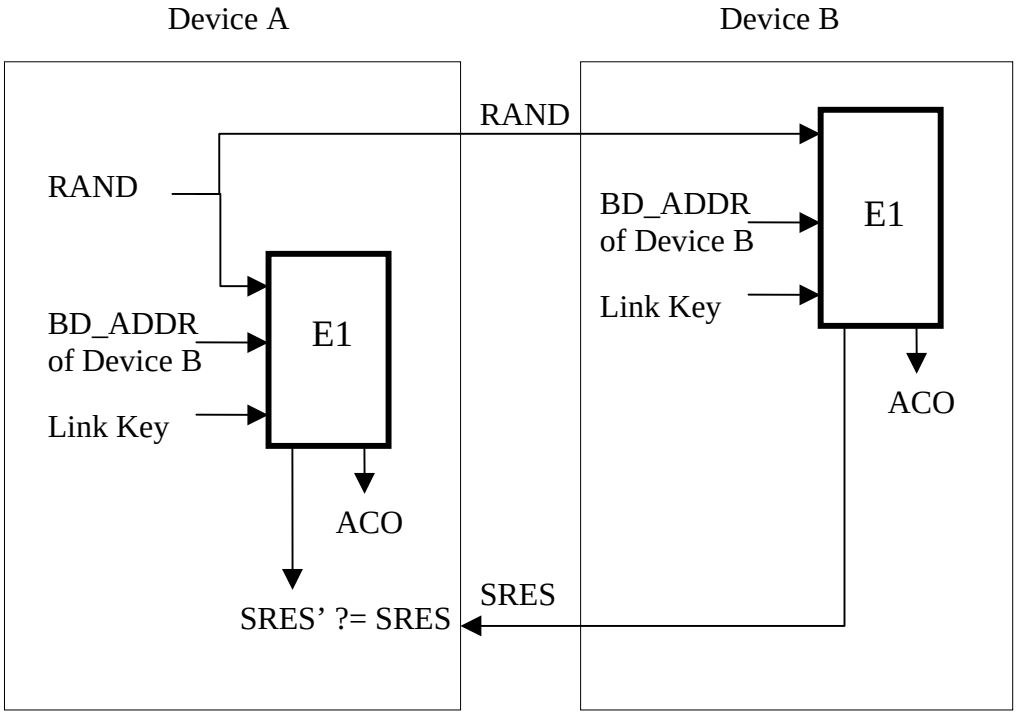

Figure 2: Authentication of Device B by Device A

5.2.2.1 Data Encryption

Bluetooth devices can perform data encryption using a stream cipher based on Linear Feedback Shift Registers (LFSR). The stream cipher generates a key stream that is used by the sender to encrypt the payload field of each packet using the one-time pad technique (the ciphertext is obtained as a result of the bit-wise exclusive-or operation performed on the payload and the key stream). The recipient of the packets decrypts the encrypted payload field of each packet by generating the key stream and combining it with the encrypted payload fields based on the one-time technique. The stream cipher is initialized by both communicating devices with the device address of the master device, the value of the shared *encryption key* and the clock of the master device. The stream cipher is resynchronized for each payload using the master's clock, a new key stream is thus generated to encrypt each payload.

5.2.2.2 Security Evaluation

The Bluetooth security architecture suffers from some weaknesses in the key management scheme. The main concern is the weakness of the key generation process for the initialization key. The initialization key is derived from a random number and a secret PIN whereby the only secret is the PIN. Due to limited capability of human memory the PIN typically is chosen as a number with at most 6 digits. A 6-digit secret can easily be retrieved by exhaustive search. Another exposure exists when a device's unit key is used as the link key. If a device's unit key is used as the link key for the purpose of parallel or

subsequent communications between this device and several other devices, the secret unit key of the device is disseminated to several devices that might include potential intruders. Various types of attacks ranging from the impersonation of the legitimate owner of the unit key to the decryption of encrypted traffic by intruders become feasible based on the knowledge of a device's unit key by intruders.

5.3 Relevance of Security Mechanisms in the Data Link Layer

While the relevance of security mechanisms implemented in the data link layer is often argued, this question deserves careful analysis in the light of requirements raised by the two different environments in which these mechanisms can be deployed:

1. wireless extension of a wired infrastructure as the original target of 802.11 and Bluetooth security mechanisms,

2. wireless ad hoc networks with no infrastructure.

In case of 1. the main requirement for data link layer security mechanisms is the need to cope with the lack of physical security on the wireless segments of the communication infrastructure. Data link layer security is then perfectly justified as a means of building a "wired equivalent" security as stated by the objectives of WEP. Data link layer mechanisms like the ones provided by 802.11 and Bluetooth basically serve for access control and privacy enhancements to cope with the vulnerabilities of radio communication links. However, data link layer security performed at each hop cannot meet the end-to-end security requirements of applications neither on wireless links protected by 802.11 or Bluetooth nor on physically protected wired links.
In case of wireless ad hoc networks as defined in 2. there are two possible scenarios:

- managed environments whereby the nodes of the ad hoc network are controlled by an organization and can thus be trusted based on authentication,

- open environments with no a priori organization among network nodes.

The managed environment raises requirements similar to ones of 1..Data link layer security is justified in this case by the need to establish a trusted infrastructure based on logical security means. If the integrity of higher layer functions implemented by the nodes of a managed environment can be assured (i.e. using tamper-proof hardware) then data link layer security can even cover higher level security requirements raised by the routing protocol or the applications.
Open environments on the other hand offer no trust among the nodes and across communication layers. In this case trust in higher layers like routing or application protocols cannot be based on data link layer security mechanisms. The only relevant use of the latter appears to be ad hoc routing security proposals whereby the data link layer security can provide node-to-node authentication and data integrity as required by the

routing layer. Moreover the main impediment to the deployment of existing data link layer security solutions (802.11 and Bluetooth) would be the lack of support for automated key management which is mandatory in open environments whereby manual key installation is not suitable.

6.Conclusions

The need for security mechanisms that cope with the threats that are specific to the ad hoc environment has recently gained attention among the research community. In order to avoid the same problems that raised in wired networks like the Internet, security has to be taken into account at the early stages of the design of basic networking mechanisms like the data link layer and the network layer protocols. Since the correct network operation can be heavily jeopardized by threats that range from simple lack of cooperation to routing message modification, ad hoc networks without a proper defense against attacks that are specific to this new networking paradigm cannot exist.

Current efforts carried out by the research community in order to support ad hoc networks with practical security mechanisms have to cope with a challenging environment, were limitations on the battery life, the computational power and the storage resources, not to mention the lack of any fixed infrastructures, make the design of a security infrastructure very difficult.

The security mechanisms presented in this chapter are a practical response to specific problems that rise at a particular layer of the network stack. However, the proposed solutions only cover a subset of all possible threats and are difficult to integrate one with the other: as an example, the secure routing protocols analyzed in this chapter do not cope with the lack of cooperation of the nodes of the network, and are not designed to incorporate a cooperation enforcement mechanism. An exhaustive security infrastructure has to consider a wide range of attacks and has to be made of easy-to-integrate components. Furthermore, security needs may vary accordingly to different networking scenarios and the security mechanisms adopted to combat misbehaving or compromised nodes have to be flexible enough to be used in different environments. The direction that has been taken by the research community in order to support ad hoc networks with security mechanisms confirms this vision and the proposed solutions are gradually reaching a mature stage, making ad hoc networking a realistic alternative to wireless and 3G networks.

7. Bibliography

[1] P. Papadimitratos, Z. Haas, Secure Routing for Mobile Ad Hoc Networks, in proceedings of CNDS 2002.

[2] Y-C Hu, A. Perrig, D. B. Johnson, Ariadne : A secure On-Demand Routing Protocol for Ad Hoc Networks, in proceedings of MOBICOM 2002.

[3] A. Perrig, R. Canetti, D. Song, J.D. Tygar, Efficient and secure source authentication for multicast, in proceedings of NDSS 2001.

[4] A. Perrig, R. Canetti, J.D. Tygar, D. Song, Efficient authentication and signing of multicast streams over lossy channels, in IEEE Symposium on Security and Privacy, 2000.

[5] B. Dahill, B. N. Levine, E. Royer, C. Shields, ARAN: A secure Routing Protocol for Ad Hoc Networks, UMass Tech Report 02-32, 2002.

[6] Y-C Hu, D. B. Johnson, A. Perrig, SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks, in the Fourth IEEE Workshop on Mobile Computing Systems and Applications.

[7] C. E. Perkins, P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, in proceedings of SIGCOMM 1994.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y-C Hu, J. G. Jetcheva, A performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, in proceedings of MOBICOM 1998.

[9] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, Scenario-based Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks, in proceedings of MOBICOM 1999.

[10] A. Perrig, Y-C Hu, D. B. Johnson, Wormhole Protection in Wireless Ad Hoc Networks, Technical Report TR01-384, Dep. Of Computer Science, Rice University.

[11] P. Michiardi, R. Molva, Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks, in proceedings of European Wireless Conference, 2002.

[12] D. B. Johnson, D. A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[13] Charles Perkins, Ad hoc On Demand Distance Vector (AODV) Routing, Internet draft, draft-ietf-manet-aodv-00.txt.

[14] L. Buttyan, J.-P. Hubaux, Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks, Technical Report DSC/2001/001, Swiss Federal Institute of Technology -- Lausanne, 2001.

[15] S. Buchegger, J.-Y. Le Boudec, Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks, in proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing.

[16] S. Buchegger, J.-Y. Le Boudec, Performance Analysis of the CONFIDANT Protocol, in proceedings of MobiHoc 2002.

[17] S. Marti, T. Giuli, K. Lai, and M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in proceedings of MOBICOM 2000.

[18] P. Michiardi, R. Molva, Core: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks, IFIP - Communication and Multimedia Security Conference 2002.

[19] P. Michiardi, R. Molva, Game Theoretic Analysis of Security in Mobile Ad Hoc Networks, Institut Eurecom Research Report RR-02-070 - April 2002.

[20] H. Yang, X. Meng, S. Lu, Self-Organized Network-Layer Security in Mobile Ad Hoc Networks.

[21] S. Capkun, L. Buttyan and J-P Hubaux, Self-Organized Public-Key Management for Mobile Ad Hoc Networks, in ACM International Workshop on Wireless Security, WiSe 2002.

[22] P. Zimmermann, The Official PGP User’s Guide. MIT Press, 1995.

[23] M. Reiter, S. Stybblebine, Authentication metric analysis and design, ACM Transactions on Information and System Security, 1999.

[24] H. Luo, S. Lu, Ubiquitous and Robust Authenticaion Services for Ad Hoc Wireless Networks, UCLA-CSD-TR-200030.

[25] A. Shamir, How to share a secret, Communications of ACM 1979.

[26] IEEE 802.11b-1999 Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher speed Physical Layer (PHY) extension in the 2.4 GHz band

[27] “Specification of the Bluetooth System”, Bluetooth Special Interest Group, Version 1.1, February 22, 2001,
http://www.bluetooth.com/pdf/Bluetooth_11_Specifications_Book.pdf

[28] “Applied Cryptography”, Bruce Schneier, John Wiley & Sons, 1996

[29] “Using the Fluhrer, Mantin, and Shamir Attack to Break WEP”, Stubblefield, Loannidis, and Rubin, AT&T Labs Technical Report 2001