We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

10

APALLS: A Secure MANET Routing Protocol

Sivakumar Kulasekaran and Mahalingam Ramkumar
Mississippi State University
USA

1. Introduction

In infra-structured networks dedicated routers relay packets between network hosts. In contrast, in mobile ad hoc networks (MANET) nodes are simultaneously network hosts and routers. Due to their reduced dependence on communication infrastructure MANETs have useful applications in scenarios where it may be impractical to set up expensive communication infrastructure, and in scenarios involving failure of communication infrastructure.

MANET routing protocols are rules to be followed by every mobile node to cooperatively discover optimal paths and route packets between end-points (source and destination nodes). In this chapter we restrict ourselves to the dynamic source routing (DSR) protocol (Johnson & Maltz, 1996). In DSR paths between end-points are established by flooding route-request (RREQ) packets originating from the source, in response to which route-response (RREP) packets are sent along the reverse path. The original DSR protocol (Johnson & Maltz, 1996) implicitly assumes that all nodes will abide by the rules. The presence of nodes that do not adhere to the rules, either deliberately, or due to malfunctioning, can have a deleterious effect on the MANET subnet. Secure MANET routing protocols strive to improve the reliability of the routing process under the presence of non cooperative nodes.

1.1 Securing DSR

Non confirming nodes can inflict a wide variety of passive and active attacks on DSR. Two basic tools employed to thwart attacks are i) mandating cryptographic authentication; and ii) monitoring neighbors to estimate their trustworthiness. A mechanism for cryptographic authentication is also a prerequisite for monitoring. Bootstrapping cryptographic authentication mechanisms mandates an authority trusted by all nodes.

1.1.1 Role of the trusted authority

The trusted authority (TA) provides secrets to nodes, thereby conferring upon them the eligibility to take part in ad hoc subnets. The TA may also revoke the privileges of some nodes, by disseminating revocation lists. The *network size N* is the total number of nodes afforded the privilege of participating in MANET subnets. Any random subset of the N eligible nodes, say N_s nodes (which may accidentally be in the same geographical region), can form a connected ad hoc subnet.

For retaining the most compelling advantage of MANETs (their reduced infrastructural support) it is desirable that the TA is *off-line*. In other words, subnets disconnected from the TA, or any other form of infrastructural support, should still be able to carry out their tasks.

In (Sanzgiri et al., 2002) such MANET networks, where the infrastructural support is required only for predistribution of keys, are referred to as *managed open* networks.

1.1.2 Resisting attacks

Active attacks involve modifying routing packets in ways that violate the prescribed protocol. For resisting active attackers secure routing protocols will require features to *detect* inconsistencies in routing information resulting from active attacks, *identify* perpetrators responsible for such inconsistencies, and have mechanisms in place to limit the role of such nodes. Effective strategies are also required to *deter* active attacks in the first place. For example, the ability to obtain non repudiable proof of active attacks can be an effective deterrent. Such proofs can be submitted to the TA and lead to revocation of such nodes.

Passive attackers may attempt to eavesdrop on messages exchanged between end-points or take selective part in routing. Submitting non-repudiable proof may not be possible[1] for passive attacks involving selective participation. Nevertheless, effective strategies are required to promote self-less participation.

Several secure routing protocols (Abusalah et al., 2008) have been proposed in the literature. Some focus on cryptographic authentication (Hu et al., 2005)-(Capkun and Hubaux., 2003), taking into consideration the resource constraints inherent to battery operated mobile devices. Some have suggested strategies like listening in the promiscuous mode (Marti et al., 2000)-(Marshall et .al, 2003), unambiguously identifying misbehaving nodes (Burmester et al., 2003)- (Awebuch et al., 2002), assigning trust metrics to nodes, and employing such metrics to advantageously influence the routing process.

1.2 Contributions

The contribution of this chapter is APALLS, a secure routing protocol based on DSR. Some of the shortcomings of current protocols that are addressed by APALLS are as follows:

1. Non-repudiable authentication, while necessary, is *not* sufficient for obtaining non-repudiable proof of misbehavior. While the importance of non repudiable authentication is well understood (Sanzgiri et al., 2002),(Sun et al., 2007), investigation of issues in obtaining non repudiable proof of misbehavior (in the context of adhering to a MANET routing protocol) has not received attention thus far. To our knowledge, APALLS is the first protocol which addresses this issue.
2. Protocols that focus on monitoring strategies (Marti et al., 2000)-(Marshall et .al, 2003) have predominantly ignored the relevance of cryptographic authentication. Likewise, protocols that focus on cryptographic authentication ignore strategies for monitoring. Comprehensive routing protocols should productively employ both.
3. While many key distribution strategies have been proposed for ad hoc networks, they have not considered realistic network models. The choice of schemes in APALLS are driven by realistic models for large scale MANET deployments.

APALLS[2] borrows some features from Ariadne (Hu et al., 2005), a popular secure extension of DSR. Ariadne does not prescribe strategies for monitoring, and consequently, does not possess mechanisms to address passive attacks involving selective participation. The cryptographic authentication strategies in Ariadne permit the source or destination to detect inconsistencies

[1] While a packet signed by a node B can be used by A to demonstrate that B violated the protocol, it is arguably infeasible for a node A prove to some third party that B did *not* forward a RREQ/RREP.

[2] Ariadne with Pairwise Authentication and Link Layer Signatures.

in path advertisements in (RREQ or RREP packets) resulting from active attacks. However, Ariadne does not attempt to *identify* the reason for the observed inconsistency, namely, the perpetrator responsible for the inconsistency. An implication of the fact that perpetrators remain unidentified is that an attacker faces *no risk* in carrying out (several types of) active attacks. Such attacks, which can reduce the efficacy of the path discovery mechanism, can be carried out repeatedly due to lack of deterrents.

APALLS provides a severe deterrent for active attacks, the threat of revocation from the network, as non repudiable proof can be submitted to the off-line TA. APALLS also recognizes that the process of revocation, involving submission of verifiable proof to the TA (when ever the node submitting the proof has access to the TA), followed by network wide dissemination of revocation lists by the TA, will not yield immediate relief from the active attacker(s) in ad hoc subnets. For this purpose, APALLS includes features to route around nodes suspected of misbehaving. APALLS also includes elements to keep an effective watch on neighbors to address passive attacks involving selective participation. Nevertheless, due to careful choice of cryptographic primitives, APALLS incorporates all these features without placing unreasonable demands on the capabilities of nodes.

The rest of this chapter is organized as follows. In Section 2 we provide an overview of DSR and Ariadne. In Section 3 we outline a generic managed-open MANET model consisting of an off-line trusted authority (TA). The threat model and the intended goals of APALLS are enumerated in Section 3.1. This is followed by a description of mechanisms for bootstrapping trust relationships between nodes of the network.

Section 4 provides the description of the APALLS protocol. The security analysis of APALLS, including rationale for the choices made, are described in Section 5. A discussion of related work can be found in Sections 6.1 and 6.2. Conclusions are offered in Section 6.3.

The following notations are used in this chapter

1. $A, B, C \cdots$ (upper case alphabets) represent unique identities of nodes;
2. $\Sigma_A = \langle M \rangle_A$ - digital signature of A for a message M.
3. $h()$ - a cryptographic second pre-image resistant hash function (not necessarily colission resistant);
4. $h(M, K)$ - hashed message authentication code (HMAC) for a message M based on a key K.
5. $C = K[P]$ encryption of a plain-text[3] P using a key K and some block cipher like AES/DES;
6. $P = K^{-1}[C]$ decryption of cipher-text C using key K.
7. $K_A^0 \cdots K_A^{L-1}$ - hash-chain of length L with commitment K_A^L, where $K_A^i = h(K_A^{i-1}), 1 \leq i \leq L$.

2. Background

Dynamic source routing (DSR) is an on-demand protocol where a node S desiring to find a path to a node T broadcasts a route-request (RREQ) packet indicating the source S, sequence number q, destination T, and a hop-limit n_h. RREQ packets are flooded. In general, every node in the same connected subnet will receive one RREQ (with the same source and sequence number) from each neighbor, but will rebroadcast only one (usually the first RREQ received). Every node rebroadcasting an RREQ inserts its identity.

[3] If the plain-text P is larger than the block size it is assumed that some appropriate mode of operation like cipher-block-chaining (CBC) is used, and the initial value is prepended to the cipher-text C.

For an RREQ from a source S, intended for a destination T, traversing a path $(A,B,C,\ldots)$

$$\begin{aligned}
\mathcal{Q}^{S}_{(q,S)} &= \mathcal{Q}_{(q,S)} = [S,q,T,n_h] \\
\mathcal{Q}^{A}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)},(A)] \\
\mathcal{Q}^{B}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)},(A,B)] = [\mathcal{Q}^{A}_{(q,S)},(B)] \\
\mathcal{Q}^{C}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)},(A,B,C)] = [\mathcal{Q}^{B}_{(q,S)},(C)], \\
&\vdots
\end{aligned} \tag{1}$$

where the $\mathcal{Q}^{X}_{(q,S)}$ represents all RREQ fields relayed by a node X, and received by the destination.

When the RREQ packet reaches the destination[4], it contains the fields $\mathcal{Q}_{(q,S)}$ specified by the source, and the list of all nodes in the path traversed by the RREQ. The destination sends a route-response (RREP) packet along the reverse path. The source and destination may in general discover multiple paths as the destination can receive one RREQ from every neighbor.

2.1 Attacks on DSR

Attacks on DSR can be broadly classified into passive, semi-active, and active attacks. Passive attackers may perform eavesdropping on application data packets exchanged between nodes, or take selective part in the network. Semi-active attackers may act as invisible relays to create misrepresentations of the network topology. Such an attacker positioned between two nodes A and B (who are out of each other's range) may simply echo packets broadcast by A such that B can receive such packets. However by echoing RREQ packets and not echoing RREP packets, the attacker can cause the route discovery to fail. Interconnected invisible relays that are physically well separated can create *worm-holes* (Hu et al., 2001) to misrepresent the subnet topology. By periodically turning on and off such worm-holes they can create rapid changes to the perceived topology, thereby inflicting significant bandwidth overhead.

Active attackers intentionally modify routing packets in violation of the protocol - for example, by inserting nonexistent nodes in the path, or deleting nodes that are actually in the path, or modifying the values inserted in the RREQ by the source and/or other upstream nodes. DSR is also susceptible to rushing attacks (Hu et al., 2003) which result from the ability of an attacker to forward RREQ packets with the intent of either creating sub-optimal routes, or even causing complete failure of the route establishment process. As each node forwards only one RREQ packet, a rushed bad packet can *preempt* other good packets.

Two basic tools employed to thwart attacks attacks are i) mandating cryptographic authentication; and ii) monitoring neighbors to estimate their trustworthiness. As an important pre-requisite for monitoring is the need to know *who* is being monitored[5], cryptographic authentication is necessary for the ability to monitor.

2.2 Cryptographic authentication

Facilitating cryptographic authentication schemes requires a trusted authority (TA) who conveys secrets to every node and/or public values associated with every node to every

[4]While in the original DSR even intermediate nodes with the knowledge of a path to the destination can invoke an RREP, in most secure DSR extensions only the destination is allowed to do so.

[5]A node A observing the behavior of a neighbor claiming-to-be-B should be able to ascertain that the observed node is indeed B before A can make any inferences about the B's trustworthiness.

node in the network. Such values provided by the TA are necessary for computing verifiable *cryptographic authentication tokens*. As packets without verifiable cryptographic authentication will be ignored, the TA confers the eligibility for nodes to take part in MANET subnets.
Cryptographic authentication can be classified into mutual (or one-to-one) authentication and broadcast (one-to-many) authentication. A secret K_{AB} known only to A and B can be used by A and B for computing and verifying pairwise authentication tokens. For a mesage M the token is typically a hashed message authentication code (HMAC) $T_{AB}(M) = h(M, K_{AB})$. Schemes for pairwise authentication thus rely on key distribution schemes that facilitate pairwise secrets between all node-pairs.
An one-to-many authentication token $T_A(M) = (R_A, M)$ for a message M can be *created* only by a node with access to a secret key R_A. The token can be verified using a public counterpart U_A of the secret R_A. The verification function $f_{ver}(M, T_{AM}, U_A)$ provides a binary output (pass or fail). If the verification test passes, the verifier can conclude that only an entity with the private counterpart of U_A could have computed the token $T_A(M)$. In addition, if it is possible for the verifier to establish that the public value U_A is indeed associated with a node A, the verifier can conclude that the token was created by A. Facilitating such schemes calls for the TA to distribute authentic public values associated with every node to every node.

2.2.1 MANET layers

In the context of DSR, "application layer" cryptographic mechanisms are required for end-points to protect the privacy and integrity of data exchanged between them. "Link layer" mechanisms are required for authentication of neighboring nodes. "Network layer" mechanisms are required for authentication of intermediate nodes that take part in relaying packets between end-points.
How end-points choose to protect application data should be left to the end-points themselves; ideally, such strategies should *not* not be under the control of the TA. However, strategies for link layer and network layer authentication *should* be promulgated by the TA.

2.3 Ariadne

In Ariadne the end-points share an application layer secret. The authentication strategies facilitated by the TA include i) a sequence-number hash chain to limit the number of RREQs that can be sent by any node; ii) TESLA broadcast authentication (Perrig et al., 2001) for authentication of intermediate nodes by an end-point; and iii) a network-wide secret at the link layer to deny access to external nodes.
In the sequence-number hash chain of a node A, $\mathcal{S}_A = \{R_A^0, R_A^1, \ldots, R_A^{n_r}\}$, $R_A^i = h(R_A^{i-1})$ for $1 \leq i \leq n_r$, and the commitment $R_A^{n_r}$ is made known to every node by the TA. Along with an RREQ with sequence number q initiated by A the value $R_A^{n_r - q}$ is released from its sequence-number hash chain.
In the TESLA hash chain of A, $\mathcal{H}^A_{t_0,\Delta} = \{K_A^0, K_A^1, \ldots, K_A^{L-1}, K_L^A\}$ with commitment K_L^A, Δ is a time-interval (for example, 1 second), and t_0 is an absolute value of time (for example, Dec 1, 2009, 0200:00, GMT). The parameters associated with A's chain (commitment K_L^A, Δ and t_0) are made known to all nodes by the TA. This TESLA chain can be used by A only for a limited segment of time (between t_0 and $t_0 + (L-1)\Delta$).
A is expected to keep the key K_A^{L-i} in its chain private at least till time $t_i = t_0 + i\Delta$. This key can be used for authenticating a value M by appending a HMAC $h(M, K_A^{L-i})$, provided the HMAC reaches potential verifiers *before* time t_i. Once K_A^{L-i} is made public (*after* time t_i) the

verifier can i) verify the TESLA HMAC; and (if consistent) ii) repeatedly hash K_A^{L-i} (i times) and verify that the result is indeed K_L^A. The verifier can then conclude that the HMAC was computed by A (as only A had access to the value K_A^{L-i} at time t_i).

2.3.1 RREQ and RREP

The steps involved in RREQ propagation from a source S to a destination T through a path $(A,B,C,\ldots)$ are depicted in Table 1 (left column). $\bar{K}_{ST}$ is the application layer shared by the end-points. The RREQ fields $\mathcal{Q}^S_{(q,S)}$ specified by the source includes a time t_i before which the destination should receive the RREQ, and a value $R_S^{n_r-q}$ from S's sequence number chain. Intermediate nodes and the destination will process the RREQ only if i) (S,q) is fresh (the node had not previously seen an RREQ from S with a sequence number q or higher); ii) hashing $R_S^{n_r-q}$ q-times yields the commitment $R_S^{n_r}$ of S; and iii) the RREQ is received before time t_i.
Apart from the fields $\mathcal{Q}^S_{(q,S)}$ which are carried forward all the way to the destination, the source S broadcasts a value β_S which is intended only for neighbors. The value β_S is chosen such that the destination T (which shares the secret $\bar{K}_{ST}$ with the source) can also compute β_S. A node A downstream of S appends two values before rebroadcasting the RREQ as $\mathcal{Q}^A_{(q,S)} = [\mathcal{Q}^S_{(q,S)}, (A, M_A)]$, where M_A is a TESLA HMAC computed using a value from A's TESLA chain which will remain A's secret till time t_i. In addition, a per-hop hash value $\beta_A = h(\beta_S, A)$ is also broadcast by A. Similarly, a node B downstream of A broadcasts $\mathcal{Q}^B_{(q,S)} = [\mathcal{Q}^A_{(q,S)}, (B, M_B)]$ and $\beta_B = h(\beta_A, B)$. Unlike TESLA HMACs, the per-hop hashes are not carried forward (they are intended only for neighbors).
The destination T computes β_S (as computed by the source) and verifies that the per-hop hash submitted by the last node in the path is consistent with the list of nodes in the path. The per-hop hash is intended to prevent node deletion attacks. Without this, a node C can simply remove the values (B, M_B) in the RREQ, and thus illegally delete B from the path. With the per-hop hash, to trick the destination into accepting a path $(A,C,\ldots)$ as valid, C will need access to the per-hop hash β_A which is privy only to neighbors of A (and C is not one). The destination will invoke an RREP only if the per-hop hash is consistent. The RREP includes all fields of the RREQ packet and is authenticated to the source using a HMAC (based on secret $\bar{K}_{ST}$).
After time t_i, the destination relays the RREP along the reverse path. Every node releases the value from the TESLA chain used for computing the TESLA HMACs during the forward path. At the end of the reverse path the source i) verifies the TESLA HMACs, and that ii) the TESLA keys are consistent with the time t_i and their respective commitments. Mandating authenication prevents illegal node insertions. In Ariadne node deletion attacks can be detected by the destination at the end of the forward path; node insertion attacks can be detected at the end of the reverse path, by the RREQ source.

3. APALLS: application model and goals

As an application of MANETs consider a (fictitious) network operator, Tingular, who desires to provide subscribers with a wide variety of services with minimal investment in infrastructure. Tingular subscribers can find other Tingular subscribers within range and form multi-hop MANET subnets. Nodes (Tingular subscribers) in a connected subnet can exchange messages with each other. Apart from communicating with other nodes in the subnet, any node with

wide-area connectivity (for example, access to the Internet) can act as a gateway and extend Internet access to all other nodes in the subnet.
The total number of Tingular subscribers at any time t, say $N(t)$, can be of the order of several millions (and varies with time t as new subscribers may join, and some may leave the Tingular network). Any small subset of current subscribers who happen to be in a geographical region like a mall, or an airport, can come-together to form a Tingular subnet (a small subnet may have just two nodes). Several Tingular subnets may operate simultaneously at different locations. While some subnets may have one or more nodes with wide-area network access, some may be completely isolated from the rest of the world. Tingular subnets will thus be able to function even during periods of natural disasters, when other communication infrastructure fail.
The main tasks to be performed by Tingular for *managing* the network are

1. to specify rules (the secure routing protocol) to be followed (a one-time process);
2. induct subscribers into the Tingular network, by providing them with secrets, possibly in exchange for a small subscription fee (performed once for every node inducted into the Tingular network); and
3. (periodically) revoke subscribers who violate rules, or do not renew their subscriptions.

Inducting a subscriber can be as simple as providing a SIM card (subscriber identity module) to the subscriber, preloaded with the secrets necessary for the node to communicate with other Tingular subscribers, and thus take part in any Tingular subnet. A "Tingular node" can be any WiFi enabled general-purpose computer (hand-held, laptop or desktop) which can house the SIM card, and can run Tingular software (which dictates the rules to be followed by Tingular nodes). Revoking nodes can be through periodic dissemination of revocation lists (posted in Tingular's website). Due to the modest investment required, Tingular can afford to provide useful services for a low subscription fee.

3.1 Threat model and goals

Some Tingular nodes in a subnet may engage in active attacks. Some such attacks may result from mere malfunctioning of nodes. Some may be perpetrated by Tingular nodes under the control of attackers. One goal of APALLS is to obtain non repudiable proof of active attacks. Such proofs can be submitted to the TA at a convenient time (as access to the TA may not be available from some ad hoc subnets), perhaps leading to revocation of such nodes from the Tingular network.
The revocation process will involve i) submission of proof to the TA, ii) verification of such proofs by the TA, and iii) dissemination of revocation lists by the TA. While the threat of revocation can be an effective deterrent, the process of revocation may not provide immediate relief from attackers in the subnet. For this purpose, APALLS will include explicit features to improve the chance of finding paths free of suspected active attackers.
Passive attacks include acts like not forwarding RREQ packets, not accepting RREP or data packets, etc. Selfish subscribers may not desire to take part in the subnet unless they require to communicate with another node. There may also exist other external attackers (who may not be Tingular nodes) performing semi-active attacks. While obtaining non repudiable proof of misbehavior is not possible for passive and semi-active attacks, APALLS aims to have tangible measures for promoting self-less behavior, and "living with" semi-active attackers.

3.2 Key distribution for APALLS

The operator of the Tingular network employs a web-server, with a certified trustworthy computer (for example, a cryptographic co-processor) at the back end. The trustworthy module is the TA. Tingular also possesses a facility for securely preloading SIM cards with secrets provided by the TA.

The TA i) generates an asymmetric key pair (R_{TA}, U_{TA}); and ii) chooses a master secret μ. Potential subscribers create a Tingular user account. On payment of the subscription fee subscribers receive a Tingular SIM card with some secrets.

The identity A assigned to a subscriber is of the form $A = [Q_A \parallel A']$ where Q_A is a serial number, and A' may be the user name. The node A is also issued an asymmetric key pair (R_A, U_A), and a certificate $C_A = E(R_{TA}, A \parallel U_A \parallel T_e)$, where T_e is time of expiry of C_A. The certificate can be decrypted any any node with access to the public key U_{TA} as $[A \parallel U_A \parallel T_e] = D(U_{TA}, C_A)$. The specifics of the functions $E()$ and $D()$ depend on the type of asymmetric primitive employed.

The subscriber sequence number is issued in a chronological order. The sequence number for the first subscriber is 1. A node A with (say) $Q_A = 1{,}000{,}000$ is the millionth subscriber. The values included in the SIM card provided to A are

1. a secret $K_A = h(\mu, A)$;
2. the private key R_A;
3. the public key of the TA, U_{TA}; and
4. the certificate C_A.

In addition, A receives $Q_A - 1$ public values of the form

$$P_{AX} = h(K_A, X) \oplus h(K_X, A), \forall X = [Q_X \parallel X'] | Q_X < Q_A,$$

where a specific $X = [Q_X \parallel X']$ represents the identity assigned to a subscriber inducted *before* A (or $Q_X < Q_A$). The public values could be downloaded from Tingular web site or mailed by Tingular (in a flash card / optical disc) over the postal network. The millionth subscriber A will receive 999,999 such public values.

The shared secret K_{AB} between two subscribers $A = Q_A \parallel A'$ and $B = Q_B \parallel B'$ is computed as follows. If $Q_A < Q_B$ then B has access to the public value P_{AB} (and A does not). The secret K_{AB} is computed as

$$K_{AB} = \begin{cases} h(K_A, B) & \text{by } A \\ h(K_B, A) \oplus P_{AB} & \text{by } B \end{cases} \quad (2)$$

The scheme described above for facilitating pairwise secrets is based on the modified Leighton-Micali scheme (MLS) (Ramkumar , 2008). If the public values (and consequently, the pairwise secrets) are 80 bits long, the ten-millionth subscriber will require about 100 MB of storage (perhaps in a flash card plugged into the hand-held computer). The maximum network size is not a hard limit. Newer subscribers (with larger sequence numbers) will just need more storage for public values.

If unlimited network sizes are desired, scalable KPSs are viable options (see Appendix 8.1).

4. The APALLS protocol

Any subscriber, after receiving the secrets and the required public values, can take part in any ad hoc subnet created by any subset of Tingular subscribers.

4.1 Joining a subnet

A subscriber A sends a probe $[A,c]$ (where c is a randomly chosen challenge) to determine other subscribers within range. A node B in the neighborhood responds with $[B,A,K_{BA}(c)]$. In general, the node A may receive one response from every neighbor within the reliable delivery neighborhood[6] (RDN) of A. Let us assume that A receives responses from 3 nodes X,Y and Z in its RDN. Node A then chooses a random one-hop group secret G_A and broadcasts individual encryptions of the secret as $[X,Y,Z,K_{AX}(G_A),K_{AY}(G_A),K_{AZ}(G_A)]$ to its neighbors. In response, A's neighbors X,Y,Z are expected to send their respective one-hop group secrets (G_X,G_Y,G_Z) to A.

Every packet sent/relayed by a node A is encrypted with the secret G_A. In other words, every node enforces a *private logical neighborhood* (PLN) (Sivakumar and Ramkumar, 2008). The PLN of a node A is (in general) a subset of nodes in the RDN of A. Node A explicitly invites some nodes into its PLN by providing a secret G_A individually to each node. If a node A, with nodes X, Y and Z in its PLN, suspects X of i) violating the protocol, or ii) acting in a selfish manner, or iii) suspect a semi-active attacker between A and X, then A can simply cut-off X from its PLN by providing a new secret to Y and Z (and withholding the secret from X). In a scenario where A has provided its PLN secret G_A to node X, but X did not send its PLN secret G_X to node A (or X did not accept A into its PLN), node A will eject X from its PLN during the next broadcast by A - which is encrypted using a new PLN secret $G_A{}'$, and the secret $G_A{}'$ conveyed only to nodes that A desires to retain in its PLN (and possibly other nodes that A desires to induct into its PLN). Thus, the PLN secret of A is updated whenever a node leaves the PLN (or is ejected by A from its PLN).

Every transmission by A is monitored by all nodes in the PLN of A, and checked for consistency with the protocol. Nodes that are observed to violate rules face the risk of being ejected from the PLN of the observer. Every node also maintains a revocation list periodically disseminated by the network operator. The revocation list can be a list of sequence numbers, signed using the TA's private key. Nodes are expected check their revocation lists before accepting a node into their PLNs. Nodes also broadcast their public key certificates to their PLN neighbors. The certificates are decrypted, and the authenticated public key of neighbors are cached.

4.2 RREQ propagation in APALLS

Table I depicts the sequence of operations involved in RREQ propagation from a source S to a destination T, through a path $(A,B,C,\ldots)$. To facilitate comparison with Ariadne the sequence of operations performed in Ariadne are also shown in the left column of Table I.

4.2.1 RREQ source S

The RREQ $\mathcal{Q}^S_{(q,S)}$ from source S specifies a maximum hop-count n_h. The RREQ can (optionally) include a list of nodes (BL) black-listed by the RREQ source. The RREQ $\mathcal{Q}^S_{(q,S)}$ includes the digital signature Σ_S of the source. In addition, the broadcast by S includes a per-hop hash β_S (as in Ariadne) intended only for nodes in the PLN; In Table I values which are not carried forward (and intended only for nodes in the PLN) are shown enclosed in flowered braces.

[6]The RDN of A includes all nodes with which A has confirmed that bi-directional links exist.

The PLN secret is used for encrypting all transmissions to neighbors. Every transmission by a node X is prepended[7] with the identity X of the node (in the clear) to enable the receiver to determine that the secret G_X should be used for decrypting the packet.

4.2.2 Intermediate nodes

Intermediate nodes verify the signature of the RREQ source. The public key certificate C_S required for verification of the signature Σ_S of the source can be included in the first RREQ sent by S in the subnet. Certificates are cached by other nodes in the subnet. If a node reviving the RREQ from S does not have access to C_S, it requests the upstream node to provide the certificate. Nodes will not rebroadcast the RREQ until they gain access to the public key of the source. If an intermediate node is included in the list BL, it simply ignores then RREQ. If an intermediate node receives an RREQ forwarded by a node in the list, the RREQ is ignored.

A neighbor A downstream of S decrypts the broadcast from S (using the PLN secret G_S provided by S) and verifies the signature Σ_S. Every intermediate node appends three values that are carried forward all the way to the destination. The fields $\mathcal{Q}^A_{(q,S)}$ broadcast by A include the triple (A, M_A, ν^S_A). M_A is a HMAC for the destination T computed using the secret K_{AT}. The value $\nu^S_A = K_{AT}[\beta_S]$ (which is not present in Ariadne) is the "encrypted upstream per-hop hash" (Sivakumar and Ramkumar, 2008).

Intermediate nodes also append 4 values which are intended only for nodes in the PLN. The 4 values broadcast by a node C are i) the per-hop hash β_C (as in Ariadne); ii) hash of the signature of the previous hop B - or $\sigma_B = h(\Sigma_B)$; iii) a value v_B, which is a one-way function of signatures appended by all nodes upstream of B; and iv) the signature Σ_C, computed over the values $\mathcal{Q}^C_{(q,S)}$, $v_c = h(v_B, \sigma_B)$, and β_C. For the first node in the path (A) the values σ, and v are not required. For notational consistency values that are not required are indicated as NULL. For nodes like B that are in the PLN of the A, the value v_A is NULL (as there is no intermediate node upstream of A).

In both Ariadne and APALLS (and more generally, any DSR-based protocol) a node with k neighbors will receive k RREQs (one from each neighbor) with the same source and sequence number. In Ariadne an intermediate nodes needs to store only one RREQ (corresponding to which an RREQ was relayed by the node), for a small duration, within which it is reasonable to expect an RREP. In APALLS an intermediate node with k neighbors will need to verify $k+1$ signatures for every RREQ (with the same source and sequence number): i) the signature of the RREQ source and ii) the signature appended by k neighbors. All k RREQs (along with the 4 one hop values) are cached for a small duration. Each node broadcasts one signed RREQ. While the signature of the RREQ source is verified by all nodes, signatures of intermediate nodes are verified *only* by PLN neighbors.

4.3 Route response

In Ariadne if any inconsistency is detected the destination simply drops the RREQ. In APALLS the destination takes some proactive steps to isolate the problem.

The destination computes $\beta_S = h(\mathcal{Q}_{(q,S)}, K_{ST})$ in the same manner computed by the RREQ source. The destination then proceeds to check the *self-consistency* and *consistency* of every node in the path. The values appended by an intermediate node C, viz., M_C and ν^B_C are deemed self-consistent by the destination T if

$$M_C = h(\mathcal{Q}^B_{(q,S)}, (C, \nu^B_C), h(K^{-1}_{CT}[\nu^B_C], C), K_{CT}). \tag{3}$$

[7] This is not shown in Table 1 to reduce notational complexity.

Ariadne	APALLS
At Node S	
$\mathcal{Q}_{(q,S)} = [S, q, T, t_i]$	$\mathcal{Q}_{(q,S)} = [S, t, T, n_h, \langle BL \rangle]$
$\beta_S = h(\mathcal{Q}^S_{(q,S)}, \bar{K}_{ST})$	$\beta_S = h(\mathcal{Q}_{(q,S)}, K_{ST}), u = h(\beta_S)$
	$\Sigma_S = \langle \mathcal{Q}_{(q,S)}, u \rangle_S$
$\mathcal{Q}^S_{(q,S)} = [\mathcal{Q}_{(q,S)}, R_S^{n_r - q}]$	$\mathcal{Q}^S_{(q,S)} = [\mathcal{Q}_{(q,S)}, u, \Sigma_S]$
$S \rightarrow * \; K_U[\mathcal{Q}^S_{(q,S)}, \{\beta_S\}]$	$G_S[\mathcal{Q}^S_{(q,S)}, \{\beta_S\}]$
At Node A	
$\beta_A = h(\beta_S, A)$	$\beta_A = h(\beta_S, A)$
	$\nu^S_A = K_{AT}(\beta_S)$
$M_A = h(\mathcal{Q}^S_{(q,S)}, \beta_A, K_A^{L-i})$	$M_A = h(\mathcal{Q}^S_{(q,S)}, \nu^S_A, \beta_A, K_{AT})$
$\mathcal{Q}^A_{(q,S)} = [\mathcal{Q}^S_{(q,S)}, (A, M_A)]$	$\mathcal{Q}^A_{(q,S)} = [\mathcal{Q}^S_{(q,S)}, (A, \nu^S_A, M_A)]$
	$\Sigma_A = \langle \mathcal{Q}^A_{(q,S)}, \beta_A \rangle_A$
$A \rightarrow * \; K_U[\mathcal{Q}^A_{(q,S)}, \{\beta_A\}]$	$G_A[\mathcal{Q}^A_{(q,S)}, \{\beta_A, \Sigma_A, NULL, NULL\}]$
At Node B	
$\beta_B = h(\beta_A, B)$	$\beta_B = h(\beta_A, B)$
	$\nu^A_B = K_{BT}(\beta_A)$
$M_B = h(\mathcal{Q}^A_{(q,S)}, \beta_B, K_B^{L-i})$	$M_B = h(\mathcal{Q}^A_{(q,S)}, \nu^A_B, \beta_B, K_{BT})$
$\mathcal{Q}^B_{(q,S)} = [\mathcal{Q}^A_{(q,S)}, (B, M_B)]$	$\mathcal{Q}^B_{(q,S)} = [\mathcal{Q}^A_{(q,S)}, (B, \nu^A_B, M_B)]$
	$\sigma_A = h(\Sigma_A), v_B = h(\sigma_A, NULL)$
	$\Sigma_B = \langle v_B, \mathcal{Q}^B_{(q,S)}, \beta_B \rangle_B$
$B \rightarrow * \; K_U[\mathcal{Q}^B_{(q,S)}, \{\beta_B\}]$	$G_B[\mathcal{Q}^B_{(q,S)}, \{\beta_B, \Sigma_B, \sigma_A, NULL\}]$
At Node C	
$\beta_C = h(\beta_B, C)$	$\beta_C = h(\beta_B, C)$
	$\nu^B_C = K_{CT}(\beta_B)$
$M_C = h(\mathcal{Q}^B_{(q,S)}, \beta_C, K_C^{L-i})$	$M_C = h(\mathcal{Q}^B_{(q,S)}, \nu^B_C, \beta_C, K_{CT})$
$\mathcal{Q}^C_{(q,S)} = [\mathcal{Q}^B_{(q,S)}, (C, M_C)]$	$\mathcal{Q}^C_{(q,S)} = [\mathcal{Q}^B_{(q,S)}, (C, \nu^B_C, M_C)]$
	$\sigma_B = h(\Sigma_B), v_C = h(\sigma_B, v_B)$
	$\Sigma_C = \langle v_C, \mathcal{Q}^C_{(q,S)}, \beta_C \rangle_C$
$C \rightarrow * \; K_U[\mathcal{Q}^C_{(q,S)}, \{\beta_C\}]$	$G_C[\mathcal{Q}^C_{(q,S)}, \{\beta_C, \Sigma_C, \sigma_B, v_B\}]$
At Node D	
$\beta_D = h(\beta_C, D)$	$\beta_D = h(\beta_C, D)$
	$\nu^C_D = K_{CT}(\beta_C)$
$M_D = h(\mathcal{Q}^C_{(q,S)}, \beta_D, K_D^{L-i})$	$M_D = h(\mathcal{Q}^C_{(q,S)}, \nu^C_D, \beta_D, K_{DT})$
$\mathcal{Q}^D_{(q,S)} = [\mathcal{Q}^C_{(q,S)}, (D, M_D)]$	$\mathcal{Q}^D_{(q,S)} = [\mathcal{Q}^C_{(q,S)}, (D, \nu^C_D, M_D)]$
	$\sigma_c = h(\Sigma_C), v_D = h(\sigma_c, v_C)$
	$\Sigma_D = \langle v_D, \mathcal{Q}^D_{(q,S)}, \beta_D \rangle_D$
$D \rightarrow * \; K_U[\mathcal{Q}^D_{(q,S)}, \{\beta_D\}]$	$G_C[\mathcal{Q}^D_{(q,S)}, \{\beta_D, \Sigma_D, \sigma_C, v_C\}]$
$\vdots$	

Table 1. RREQ (From Source S to Destination T, Sequence Number q) Propagation in Ariadne (left) and APALLS (right) over a path $(A, B, C, D, \ldots)$.

A self-consistent node C, with an upstream node B is deemed *consistent* only if the per-hop hash C claims to have received from B, viz, $h(K_{CT}^{-1}[\nu_C^B])$, matches what B claims to have broadcast, viz., $h(K_{BT}^{-1}[\nu_B^A], B)$. Verifying the consistency of C is possible only if B is found to be self-consistent, or

$$M_B = h(\mathcal{Q}^A_{(q,S)}, (B, \nu_B^A), h(K_{BT}^{-1}[\nu_B^A], B), K_{BT}). \tag{4}$$

Corresponding to every path through which the destination T receives an RREQ with *no* inconsistencies, the destination invokes an RREP of type $SUCC$. For an RREQ received through a path (A, B, W, G, H, P), the RREP is of the form

$$\mathcal{P}_q = [(u, \beta_S), \{SUCC : (A, B, W, G, H, P)\}] \tag{5}$$

The RREP is authenticated to the source using a HMAC based on the secret K_{ST}. The value u is a convenient index to the RREQ. That the RREP includes β_S (the preimage of u) informs the intermediate nodes that the destination did indeed invoke an RREP.
Some paths may contain one or more inconsistent nodes. As an example, consider a scenario where an RREQ indicates a path (A, B, C, D, E, F), and the destination determines that while D, E and F are consistent, C's consistency cannot be verified because B was not self-consistent. The RREP sent by T is then

$$\mathcal{P}_q = [(u, \beta_S), \{FAIL : ((B\lambda C), (D, E, F))\}] \tag{6}$$

The RREP is authenticated individually (using individual HMACs) for verification by the consistent nodes D, E, F and the last self-consistent node C. The special code λ indicates that an inconsistency was detected in the RREQ. Nodes that are identified as consistent (D, E and F) consider this RREP as an instruction to drop subsequent RREQs from S to T *if* they include C or B in the path (for example, if the source S sends a second RREQ after the first one times out). This is also considered by nodes D and C as a request to store RREQs they had received from their respective upstream nodes (C and B) in non volatile storage, for submission to the TA at the earliest opportunity (for example, when the node has access to the Internet).

5. Rationale and security analysis

Some significant differences between Ariadne and APALLS are i) use of pairwise secrets instead of TESLA; ii) mandating the RREQ source to append a digital signature; iii) enforcing a PLN; iv) use of an additional upstream per-hop hash; and v) a digital signature appended by every node broadcasting an RREQ, intended only for one-hop neighbors.

5.1 Choice of cryptographic authentication strategies

While Hu et al (Hu et al., 2005) preferred TESLA, Ariadne was also designed to support pairwise secrets (Ariadne-PS) or digital signatures (Ariadne-DS) instead of TESLA. In Ariadne-DS every intermediate node appends a digital signature (instead of an HMAC), which is carried forward all the way till the destination. The obvious disadvantage of using digital signatures is the overhead. This is exacerbated in Ariadne due to the fact that signatures and public key certificates appended by every node in the RREQ path have to be carried over all the way to the destination. Another disadvantage is the increased susceptibility to trivial denial of service (DoS) attacks due to the high verification complexity. An attacker can send random "signed packets" which can be recognized as bogus only after the expensive verification process.

When pairwise secrets are used the HMACs by intermediate nodes are computed as in APALLS. End-points can also readily use a secret K_{ST} for authenticating RREQ/RREP instead of relying on an out-of-network mechanism for establishing an application layer secret $\bar{K}_{ST}$. No additional values need to be released during the reverse path; the destination can detect both deletion and insertion attacks. Furthermore, issues related to the delay sensitivity of TESLA based authentication (Sivakumar and Ramkumar, 2008) can also be avoided.
Another advantage of the fact that an out-of-network mechanism is not required to establish pairwise secrets between end-points is that it opens up mechanisms for salvaging broken routes, or re-routing RREPs. In a scenario where a node D in the path between S and T detects that the path is broken, if D does not share readily a secret with S or T (in Ariadne-TESLA and Ariadne-DS), it cannot raise an RREQ to S or T to find alternate paths. In Ariadne-PS, D can do so.
The primary differences in rationale for the choice of key distribution strategies in Ariadne and APALLS stem from the differences in the assumed network model.

5.1.1 Pairwise secrets instead of TESLA

The reason Hu et al (Hu et al., 2005) preferred TESLA over pairwise secrets was that "broadcasting N commitments is easier than distributing $\binom{N}{2}$ secrets." This is indeed true if a trusted entity is available in every subnet to broadcast commitments. Broadcasting $N_s = 200$ (where N_s is the number of nodes in a specific subnet) TESLA commitments within a subnet is indeed preferable to unicasting $\binom{N_s}{2} = 19,000$ values (or unicasting 199 unique values to every node).
However, keeping in mind that one of the most compelling advantages of MANET is the ability to operate without infrastructural assistance (like a trusted entity in every subnet), for large scale dynamic MANET networks ($N(t)$ of the order of several millions) distributing TESLA commitments can be more expensive. The only practical approach may be for each node to possess a certified asymmetric key pair which is used for authenticating TESLA commitments. In Ariadne, where the TESLA HMACs can be verified only by the destination, an intermediate node X will need to release (during the RREP), a value from its TESLA chain, and in addition, i) the TESLA commitment; ii) a digital signature of X for the commitment; and iii) a certificate (issued by the TA) for the public key X.

5.1.2 RREQ signatures

In Ariadne the choice of the strategy for controlling RREQ floods is also based on the implicit assumption that a trusted entity is available in every subnet. In the absence of such an entity in each subnet, an attacker can collect sequence-number hash chain values made public in one subnet to flood RREQs in other subnets. Thus, for the assumed network model, sequence-number chains are not useful.
Furthermore, while the RREQ hash chain value provides implicit authentication to two fields (source and sequence number), it does not protect other fields like the destination. Thus, if an active attacker forwards the RREQ after changing the destination field, such an RREQ will be propagated by downstream nodes. To authenticate all RREQ fields specified by the source Hu et al suggested the use of chained one-time signature (OTS) schemes (Hu et al., 2003). In chained OTS schemes, in a chain $C_0, C_1, \ldots C_L$ the values (C_i, C_{i+1}) are keys pairs of an OTS scheme which can be used to sign the RREQ. The signer can use C_i to compute a signature which can be instantaneously verified by any node with access to C_{i+1}. The next RREQ from the source will make C_i public and use C_{i-1} to sign the message. This approach also relies on

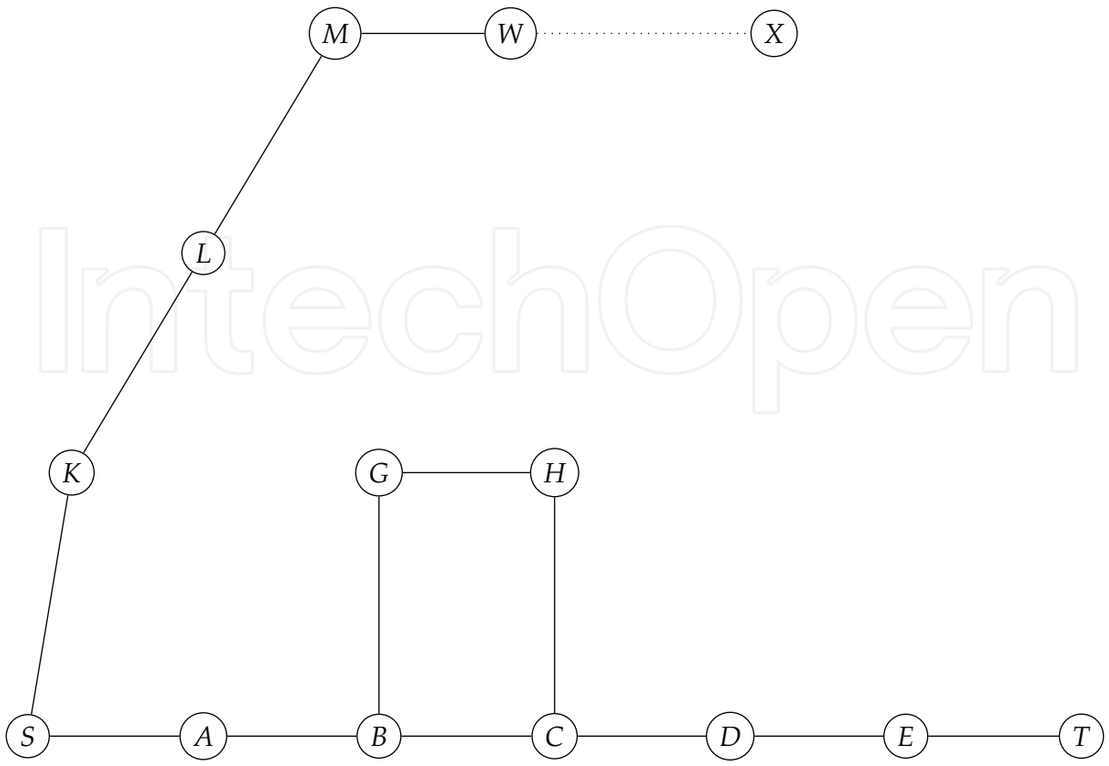


Fig. 1. Network Topology Used for Illustrations.

the same assumption (that the network is the subnet) as a value made public in one subnet can be used in another subnet.

Mandating digital signatures by RREQ source can prevent such attacks. Furthermore, as RREQs have to be authenticated, if a node S floods too many RREQs other nodes will simply drop RREQs from S in the future.

5.2 Rationale for PLN

Perhaps the most important implication of the fact that inexpensive schemes for computing pairwise secrets exist (MLS requires only one hash computation) is that it becomes practical to enforce PLNs, and derive many tangible benefits out of this ability. More specifically, enforcing the PLN is intended to achieve the following goals: i) eliminate trivial risk-free active attacks; ii) validate the assumption behind the security of the per-hop hashing strategy; iii) avoid one-way links and "links" created by semi-active attackers; iv) deter selfish behavior.

5.2.1 Trivial risk-free attacks

Consider a representative topology in Figure 1, for a specific scenario where a malicious node C receives a RREQ originating from a node S (intended for the destination T) through the path (A, B).

In Ariadne the shared network-wide secret K_U does not prevent nodes from impersonating other nodes for purposes of fooling their neighbors. For example, C, claiming to be "node C'," can relay the RREQ indicating a path (A, B, C'). All that D (which receives the packet from "node C'") can verify is that the C' has access to the network-wide shared secret K_U. While paths that include C' will be dropped by the source (in Ariadne-PS) or the destination

(in Ariadne-TESLA), such RREQs can preempt good RREQs. Furthermore, C does not face any risk of being identified.
Note that Ariadne-DS is not susceptible to such attacks as the signature appended by C can be verified by its neighbors before the RREQ is forwarded. APALLS employs two independent forms of link-layer authentication: one based on the one-hop PLN secret, and one based on digital signatures. The PLN authentication also serves as a protection against DoS attacks that could be launched due to the higher computational overhead required for verifying digital signatures. RREQ packets relayed by a node C are first decrypted using PLN secret G_C. Only if the resulting packet is a valid[8] RREQ packet will the receiver proceed to verify the signature Σ_C.

5.2.2 Protecting per hop hash

Many secure routing protocols (including Ariadne) simply assume bidirectional links. Often the justification provided for this is that "the handshake used in colission avoidance protocols ensures bidirectional links" (Kim & Tsudik., 2005). Unfortunately, this is not sufficient to prevent a node C which *can* overhear packets sent by a node B from *pretending* that it cannot (Sivakumar and Ramkumar, 2006).
When B relays a RREQ from S indicating a path (A,B) and a per-hop hash value β_B, C can wait for the RREQ to be relayed along another path, say (A,B,G,H). Now, with access to β_B (as C *can* hear B), C has the ability to remove its immediate upstream neighbor H (or G and H) from the path. Imposing a PLN is *necessary* to validate the assumption behind the security of the per-hop hashing technique - that nodes that are not neighbors cannot gain access to the per-hop hash. If C pretends to be out of the range of B, B will not include C in its PLN (thus, C will not gain access to the per-hop hash broadcast by B).

5.2.3 Semi active attacks

If a PLN is enforced a semi active attacker between B and C has to rebroadcast packets from B and the response from C when they induct each other into their PLNs. B and C may have reasons to suspect a semi active attacker when they hear an echo of their own transmissions, or if an abnormally large delay is observed in handshakes between B and C (Hu et al., 2003). Under such suspicions they will simply not induct the other node into their PLNs.

5.2.4 PLN as a deterrent

Without the ability to impose a PLN all that a node B can do to reduce the participation of a bad neighbor C is to ignore packets from C. However, B cannot prevent C from forwarding packets sent by B. The ability to cut-off neighbors in the physical neighborhood from the PLN facilitates DoS-free countermeasures to reduce the ill-effects of malicious nodes. Nodes cut off by all neighbors are effectively cut off from the subnet. Mandating a PLN can also deter selfish selective participation. A node C will have to be *inducted* into the PLNs of its neighbors before C can actually monitor traffic. Once inducted, C is pressured to participate self-lessly due to the fact that it is under constant observation by its neighbors, who may cut C off if they sense selfish participation.

[8] Valid RREQ packets may start with a magic number. In addition, for a packet from C, the last entry in the path field should be C.

5.3 Identifying and revoking active attackers

Enforcing a PLN does *not* address *all* risk-free attacks. As an illustration, consider a scenario where C receives a RREQ (from S to T) along a path (A,B), and relays the RREQ indicating a path (Q,R,C) instead, where Q and R are fictitious nodes inserted by C. Nodes downstream of C have no reason to suspect that Q and R do not exist, and B does not have access to β_S and hence $\beta_Q = h(\beta_S, Q)$ or $\beta_R = h(\beta_Q, R)$ to verify that the value β_C is indeed inconsistent.

Note that *if* C had instead advertised a path (A,R,C) with a random β_C, B (which has access to β_A) can determine that $\beta_C \neq h(h(\beta_A,R),C)$. Similarly, if C had modified any of the fields specified by the "real" upstream nodes (A and B), then B can recognize such attempts. Thus, while there are some *blatant* active attacks which can be easily be recognized by neighbors, some subtler attacks can not.

Assume that the destination receives the tainted RREQ indicating a path (Q,R,C,D,E,F,G) and a per-hop hash β_G. Assume that the actual reason for the inconsistency in the RREQ was that C had preformed an active attack. In Ariadne all that the destination can detect at this point is that "the per-hop hash β_G is inconsistent." In Ariadne-DS and Ariadne-PS T can also conclude with certainty that node G exists (as T can verify the HMAC / signature of G). However, T cannot verify the authentication appended by F as T does not access to the value β_F (which had gone into the computation of the authentication appended by F). Thus T cannot even determine if the node F actually exists in the path.

If T desires to determine *who is responsible* for perpetrating this attack, it can come to several likely conclusions: like i) G is a malicious node and every other node in the path has been maliciously inserted by G; or ii) G is a good node, but F may have maliciously inserted nodes (A,B,C,D,E) in the path; or iii) both G and F are good nodes and the node E may have inserted nodes (A,B,C,D) in the path; and so on.

In Ariadne-DS the destination can then demand all intermediate nodes (A,B,C,D,E,F,G) to produce the per-hop hash they had received from *their* upstream neighbor, which is simultaneously consistent with the signature of the upstream node (which was already included in the RREQ sent to the destnation). Now node D can produce a value β_C consistent with the signature Σ_C, and $\beta_D = h(\beta_C, D)$ consistent with D's signature Σ_C. Likewise, all nodes that had *not* violated the protocol can also do so.

However, the attacker C cannot produce a value β_R consistent with the "signature Σ_R." The obvious recourse for C is to not respond to this demand (C could just power off or leave the subnet). Now, as it is not possible to compute the value β_R (which according to C, was sent by R) from $\beta_C = h(\beta_R, C)$, one cannot deduce that Σ_R is indeed inconsistent with β_R. If C has to be convicted based on its inability to provide an "affirmative defense" (providing β_R consistent with Σ_R) it is indeed possible that an innocent D, which had suddenly crashed (and thus loses the value β_C) can also suffer the same fate.

5.3.1 Proof of active attacks in APALLS

The encrypted upstream per-hop hash ν in APALLS serves two purposes. Firstly, it makes it possible for the destination to narrow down active attackers. For example, if in the path (A,B,C,D,E,F,G) the destination is able to determine that nodes (D,E,F,G) were consistent, and C, while self-consistent, cannot be verified to be consistent (as B is self-inconsistent), the destination can narrow down the active attacker to B or C. Secondly, when used in conjunction with one-hop signatures, it facilitates unambiguous identification of active attackers, and avoids the need for nodes to provide affirmative defense.

In other words, even without carrying over all signatures (thereby saving bandwidth overhead for signatures and public-key certificates) APALLS can provide non repudiable proof of active attacks. Irrespective of the nature of the active attack, a signed packet from the attacker (stored temporarily by a neighbor, and submitted to the TA at a convenient time) can be used for this purpose.
Note that the values broadcast by C is effectively a non repudiable statement to the effect "the fields $\mathcal{Q}^B_{q,S}$, $\beta_B = K^{-1}_{CT}[\nu^B_C]$, and v_B, were broadcast by B, and verified by me (C) to be consistent with the signature of B (Σ_B), the preimage of σ_B."
When the values stored by D (the contents of the RREQ broadcast by C) are submitted to the TA, the TA takes the following steps:

1. Verify that Σ_C is consistent with $\mathcal{Q}^C_{q,S}$, β_C, σ_B and v_B;
2. Check if B is a valid node in the network; if not, C is an active attacker (C had inserted a nonexistent node in the path);
3. If B is a valid node, compute the signature $\Sigma_B{}'$ for the values $\mathcal{Q}^B_{q,S}$ and $\beta_B = K^{-1}_{CT}[\nu^B_C]$ and v_B (which according to C, were broadcast by B), and
4. Verify if $h(\Sigma_S{}') = \sigma_B$. If so, B is an active attacker (as B advertised self-inconsistent values (B, M_B, ν_B)). If not, C is the active attacker (as C had accepted a packet with an invalid signature).

If the TA has access to the private keys of all nodes the TA can simply compute $\Sigma_B{}'$. If private keys are *not* escrowed by the TA, the TA will need to request B to produce a verifiable signature $\Sigma_S{}'$ for the values $\mathcal{Q}^B_{q,S}$ and $\beta_B = K_{CT}[\nu^B_C]$ and v_B. Thus even in scenarios where the private keys are not escrowed by the TA, unlike Ariadne-DS, nodes will only need access to their private key to avoid being penalized (revoked[9]) accidentally.
A compelling advantage of escrowing private keys by the TA is that the verification of proof of attacks can be performed immediately. This is especially useful in scenarios where access to the TA is available (for example, if at least one node in the subnet has Internet access), as the revocation message (signed by the TA) can be immediately distributed within the subnet.

5.4 Routing around attackers

In scenarios where access to the TA does not exist, nodes in the subnet will have to "live with" active attackers for some (indefinite) duration. APALLS includes two strategies for improving the ability to route around nodes suspected of active attacks. The first is by using black-lists specified by the RREQ source. The second is by employing RREPs with a $FAIL$ code.
The list of nodes in S's black list can include nodes which were possibly S's neighbor at some time in the past, and observed by S to violate the protocol, or engage in selfish behavior. The list can also include nodes which have been recognized as active attackers when S was a destination node in some RREQ. That a node X is black-listed by a node S is not interpreted by other nodes to mean that "X is malicious." All this means is that the source S desires to avoid X in paths where S is an end-point. Thus, the black-list of S will only influence routing of RREQ packets in which S is the source or the destination.
The second strategy is intended to improve the success of the second RREQ that may be sent by the source S after the first RREQ times out. For instance, in a scenario where the $FAIL$ RREP indicates $[(B\lambda C), (D, E, F, G)$, during the second RREQ the nodes (D, E, F, G) will drop

[9] Any node which claims to not have access to its private key should be revoked in any case.

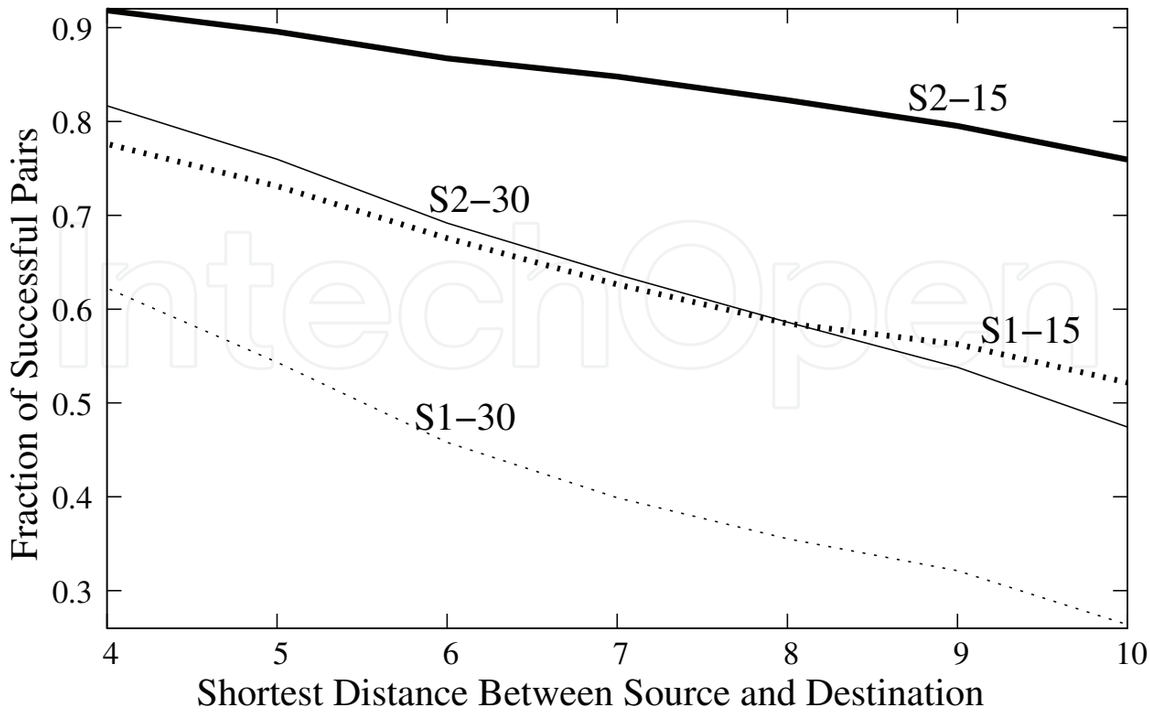


Fig. 2. Simulation results depicting the utility of the ability to narrow down the perpetrator.

RREQs that include C or B. Note that without this measure (and if the subnet topology has not changed) the second RREQ may suffer the same fate as the first RREQ.

To evaluate the benefit of this strategy simulations were performed with random realization of subnets with $N_s = 200$ nodes with uniformly distributed x and y coordinates in a square region with unit edges. The range of the nodes was chosen as 0.1 units (each node had 5 neighbors on an average). Of the $N_s = 200$ nodes, b randomly chosen nodes were labeled malicious. RREQ propagation was simulated between every pair of nodes.

Three different realizations of the network were simulated with different sets and numbers of "bad" nodes. The simulation results are depicted as fraction of node-pairs that succeed in discovering a path free of bad nodes (y-axis) vs the shortest number of hops between the pair (between which RREQ propagation was simulated) as the x-axis. RREQ propagation was simulated for over 400,000 pairs separated by hop lengths between 4 and 10. Path discovery between a pair is assumed to succeed if at least one of the established paths is free of b malicious nodes. In Figure 2 plots labelled S1 depict the success rates of first RREQs. Simulation results are shown for $b = 15$ (S1-15) and $b = 30$ (S1-30).

The plots labelled S2 indicate fraction of successful node pairs after the second RREQ (either the first or the second RREQ attempt succeeds). As can be seen from the simulation results the success rate after the second RREQ for the scenario with 30 bad nodes (S2-30) is comparable to the success of the first RREQ with just 15 bad nodes (S1-15). It is important to note that in the absence of this strategy, the second RREQ has only as much chance of succeeding as the first. Thus, in this particular instance, it can be argued that the additional upstream per-hop hash helps in realizing a two-fold improvement in resistance to malicious nodes in the subnet.

5.5 RREP authentication

In Ariadne the authentication appended by the destination for the RREP (which is verifiable only by the RREQ source) is indistinguishable from a random number for all intermediate nodes that relay the RREP. This can be exploited by attackers to send spurious RREPs over long (fictitious) paths to cause unnecessary bandwidth overhead for other nodes in the subnet. Nodes specified in the path will simply forward the RREP along the path specified. This attack is particularly dangerous in Ariadne as every intermediate node will need to release a TESLA key *and* a certificate for a commitment.

Consider a scenario where an RREQ from a source S to some destination T indicates t_i (as the upper limit before which the destination T should receive the RREQ). Assume that such an RREQ through a path (K, L, M) is heard by an attacker W. Just by overhearing any RREP packet in response to *any* RREQ (not necessarily a response for the RREQ from S) *after* time t_i, it is possible for the attacker W to harvest a preimage K_X^i corresponding to time t_i of some node X. The node X may even be many hops away from W. A malicious W can now send a fictitious RREP indicating a path (K, L, M, X) to M with a random HMAC by "destination T". All that nodes (K, L, M) can verify is that K_X^i is indeed i^{th}pre-image of K_X^0. Obviously this serves very little purpose without the ability to recognize the authentication appended by the RREP destination (which conveys the crucial information that the HMACs were received *before* time t_i). Effectively, *any* node can send such spurious RREP packets in response to any RREQ packet, impersonating some other node which may be several hops away.

In APALLS the destination includes a value β_S in the RREP which was until then known only to the source and destination. Thus, even while supercilious RREPs can be sent by nodes (which will be detected by the source as inconsistent), such RREPs can be raised only by nodes which had actually seen an RREP from the destination. Furthermore, such an attack is not worthwhile for any attacker as the RREP overhead is small in any case in APALLS.

6. Related work and conclusions

Several authors have investigated strategies for securing DSR, and mechanisms for cryptographic authentication.

6.1 Other secure DSR protocols

Papadimitros (Papadimitratos and Haas., 2002) et al propose a secure routing protocol (SRP) where only the source and destination share a secret. Marshall et al (Marshall et .al, 2003) argued that SRP cannot avoid malicious behavior by intermediate nodes during the route establishment phase, as long as the (malicious) behavior is consistent in the forward and reverse path. They also suggest techniques to mitigate issues in SRP by employing promiscuous mode of operation (Marti et al., 2000).

Kim et al (Kim & Tsudik., 2005) (SRDP) propose a general protocol for securing route discovery in DSR, where the primary deviation from Ariadne is that they strive to reduce the bandwidth overheads by aggregating the authentication appended by intermediate nodes (for Ariadne-PS and Ariadne-DS where the destination can verify authentication appended by intermediate nodes). The disadvantage of aggregating authentication is that the destination cannot verify *which* node was responsible for the inconsistency. As Ariadne does not strive to do that in any case, aggregating authentication can reduce RREQ overhead for Ariadne. However, aggregating HMACs can not be done for APALLS as it would not permit detection of self-consistency of nodes.

APALLS is an extension of an earlier work (also by the authors of this chapter) (Sivakumar and Ramkumar, 2008) which sought to improve the resiliency of Ariadne-PS. The improvements suggested in (Sivakumar and Ramkumar, 2008) include i) use of the upstream per-hop hash to narrow down active attackers; and ii) enforcing a PLN. The modifications in APALLS compared to (Sivakumar and Ramkumar, 2008) are: i) the use of one-hop digital signatures for non-repudiation; ii) mandating digital signature by the RREQ source; and iii) a modified strategy for authenticating RREPs.

6.2 Key distribution

Several key distribution schemes have been proposed in the literature for ad hoc networks. Zhou et al (Zhou and Haas., 1999) propose a key management service with distributed CA, using threshold cryptography to distribute shares of the CAs private key to several nodes. Capkun et al (Capkun and Hubaux., 2003) propose a strategy for "building secure routing from an incomplete set of security associations" (BISS), in which a combination of predistribution of keys (which facilitates only an incomplete set of pairwise secrets) and public key primitives are used. The motivation for BISS seems to be that schemes for establishing pairwise secrets between a fraction of nodes is more practical than schemes that permit *every* pair nodes to establish a secret.

Zhang et al (Zhang et al., 2005) propose the use of identity based encryption an signature (IBE / IBS) schemes for ad hoc networks. IBS schemes can reduce the bandwidth overhead for signatures as i) public keys and public key certificates are not required; and ii) the signatures are also generally smaller than (say) RSA signatures. This advantage is not compelling in APALLS as signatures are not carried forward. Unlike RSA signatures where we can reduce signature verification complexity by choosing small public exponents, IBS schemes do not have practical strategies to reduce verification complexity. High verification complexity can lead to simple DoS attacks. However, in APALLS, this is not a disadvantage as the low complexity PLN-based authentication (which is verified before signatures are verified) can prevent such DoS attacks. Thus, both the advantages and disadvantages of IBS schemes are less relevant in APALLS.

6.3 Conclusions

We have outlined a comprehensive secure routing protocol, APALLS, based on DSR. To the extent of our knowledge, APALLS is the first secure routing protocol which is designed to provide non repudiable proof of active attacks.

Non-repudiable authentication is necessary, but not sufficient to provide non repudiable proof of active attacks. In general, any active attack involves violation of the prescribed protocol. The protocol prescribes the steps that a node (say) C should take in response to a packet sent from a neighbor (say) B. For example, in distance vector based protocols, if a node B announces a hop-length of 5 to a node S, the neighbor C downstream of B is expected to announce a hop-length 6.

In a scenario where C advertises a hop-length 7, proving that C did (or did not) violate the protocol requires several *contextual* information like (for example) i) if B was indeed a neighbor of C at that time; ii) the hop count advertised by B at that time ; iii) if C did indeed process the information advertised by B (the packet broadcast by B did not suffer colission), etc.. Thus, even while some ad hoc routing protocols like ARAN (Sanzgiri et al., 2002) and Ariadne-DS employ non repudiable authentication, they do not address the issue of *how* a packet sent from a node can be used for proving an active attack. As pointed out in this chapter, even

while Ariadne-DS carries forward all signatures, it still has practical issues in providing non repudiable proof.

One of the motivations for APALLS stem from the fact that the main advantage of MANET based networks is their ability to operate without any infrastructural support. Ideally, while we would desire to eliminate even an off-line TA, this is simply not possible to do so as an authority is required to i) specify the rules (the protocol) that should be followed by every node; and ii) to boot-strap cryptographic associations between nodes.

While APALLS borrows some features from Ariadne, the major differences between Ariadne and APALLS stem from the network model. Several elements in Ariadne like i) the preference of TESLA over pairwise secrets; ii) the choice of the strategy to suppress RREQ floods; and iii) ignoring the risk of supercilious RREPs (RREP bandwidth can be high if a TA is not available in the subnet) assume the presence of a TA in every subnet. While APALLS can take advantage of access to TA (when at least one node in the subnet has access to the Internet) for quickly disseminating revocation lists, APALLS can operate effectively even in subnets that may be completely isolated from the rest of the world.

The choice of cryptographic authentication schemes in APALLS are also driven by the need to keep the overhead low. Storage is an inexpensive resource for mobile devices; any mobile device can easily afford several GBs of pluggable storage. However computational and bandwidth overheads are expensive for battery operated devices. This renders key predistribution schemes for pairwise secrets (which impose low computational and bandwidth overhead) well suited even for dynamic large scale networks.

That digital signatures appended by intermediate nodes are verified only by neighbors renders just about any scheme well suited for this purpose. More specifically, it also opens up the feasibility of non repudiable one-time signature (OTS) schemes[10] which do not require asymmetric primitives. That only neighbors need to verify the signature renders the scheme proposed by Merkle et al (Merkle, 1987) for constructing infinite OTS trees substantially more efficient. That OTS schemes require only block-cipher/ hash operations implies that even very low complexity SIM cards can perform the operations required for this purpose. Such low complexity SIM cards which need to perform only symmetric cipher operations can be realized at lower cost.

Some of the ongoing work of the authors include i) investigation of the suitability of OTS schemes; and ii) use of one-hop signatures for providing non repudiable proof of active attacks for other MANET routing protocols like AODV (Perkins et al., 2002), TORA (Park and Corson, 1997) and OLSR (Jacquet, 2001).

7. References

Johnson, P., Maltz, D. (1996). Dynamic source routing in ad hoc wireless networks, *Mobile Computing, Kluwer Publishing Company,*, ch. 5, pp. 153-181.

Sanzgiri, K., Dahill, B., Levine, N., Shields, C., Belding-Royer, E.M. (2002). A Secure Routing Protocol for Ad Hoc Networks, *Proceedings of the 2002 IEEE International Conference on Network Protocols (ICNP),* November 2002.

Abusalah, L., Khokhar, A., Guizani,M. (2008). A Survey of Secure Mobile Ad Hoc Routing Protocols, *IEEE Communications Surveys and Tutorials,* 10(4), 2008.

[10]This does *not* include chained OTS schemes which cannot be used for non repudiation as private keys are revealed eventually.

Hu, Y.C., Perrig, A., Johnson. D.B. (2005). Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks, *Journal of Wireless Networks,*11, pp 11–28, 2005.

Kim, J., G. Tsudik. (2005). SRDP: Securing Route Discovery in DSR, *IEEE Mobiquitous'05,* July 2005.

Zhang, Y., Liu, W., Lou, W., Fang, Y., Kwon, Y. (2005). AC-PKI: anonymous and certificate less public key infrastructure for mobile ad hoc networks, *IEEE International Conference on Communications (ICC'05),* Seoul, Korea, May 2005.

Zhou, L., Haas, Z. (1999). Securing Ad Hoc Networks, *IEEE Network,* 13(6), pp 24-30, 1999.

Capkun, S., Hubaux, J-P. (2003). BISS: Building Secure Routing out of an Incomplete Set of Security Associations, *In Proceedings of the Wireless Security Workshop (WISE) 2003*, San Diego, September 2003.

Marti, S., Giuli, T J., Kevin Lai., Mary Baker. (2000). Mitigating routing misbehavior in mobile ad hoc networks, *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking,* Boston,2000.

Marshall, J.,Thakur,V., Yasinsac.A. (2003). Identifying flaws in the secure routing protocol, *Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference,* 2003.

Burmester, M., Van Le, T., Weir. M. (2003). Tracing Byzantine Faults in Ad Hoc Networks, *Proceedings of Communication, Network, and Information Security (CNIS),* NY, Dec 2003.

Awerbuch, B., Holmer, D., Nita-Rotaru, C., Rubens, H. (2002). An On-Demand Secure Routing Protocol Resilient to Byzantine Failures, *ACM Workshop onWireless Security (WiSe–02),* September 2002.

Sun, J., Zhang, C., Fang, Y. (2007). An id-based framework achieving privacy and non-repudiation in vehicular ad hoc networks, *MILCOM*, 2007.

Hu, Y.C., Perrig, A., Johnson. D.B. (2001). Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks, *Rice University Department of Computer Science Technical Report TR01-384,* Dec 2001.

Hu, Y.C., Perrig, A., Johnson. D.B. (2003). Rushing Attacks in Wireless Ad Hoc Network Routing Protocols, *WiSe 2003,* San Diego, CA, September 2003.

Perrig, A., Canetti, R., Song, D., Tygar, D. (2001). Efficient and Secure Source Authentication for Multicast, *In Network and Distributed System Security Symposium,* NDSS '01, Feb. 2001.

Ramkumar, M. (2008). On the Scalability of a Nonscalable Key Distribution Scheme, *IEEE SPAWN 2008,* Newport Beach, CA, June 2008.

Sivakumar, K. A., Ramkumar, M. (2008). Improving the Resilience of Ariadne, *IEEE SPAWN 2008*, Newport Beach, CA, June 2008.

Sivakumar, K A., Ramkumar, M. (2009). Private Logical Neighborhoods for Wireless Ad Hoc Networks, *5-th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet),* Canary Islands, Spain, October 2009.

Hu, Y.C., Perrig, A., Johnson. D.B. (2005). Efficient Security Mechanisms for Routing Protocols, *Symposium on Networks and Distributed Systems Security (NDSS),* 2003.

Sivakumar, K A., Ramkumar, M. (2006). On the Effect of Oneway Links on Route Discovery in DSR,*Proceedings of the IEEE International Conference on Computing, Communication and Networks, ICCCN-2006,* Arlington, VA, October 2006.

Papadimitratos, P., Haas, Z.J. (2002). Secure Routing for Mobile Ad Hoc Networks, *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference(CNDS 2002),* San Antonio, Texas,2002.

Merkle, R.C. (1987). A digital Signature based on Conventional Encryption Function, *Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, Lecture Notes In Computer Science;* 293, pp 369 – 378, 1987.

Perkins, C., Royer,E., Das.S. (2002). Ad hoc On-demand Distance Vector (AODV) Routing, *Internet Draft, draft-ietf-manet-aodv-11.txt, Aug 2002. The 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002),* 2002.

Park, V.D ., Corson, M.S. (1997). A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks, *Proceedings of IEEE INFOCOM,* Kobe, Japan, 1997.

Jacquet, P., Mühlethaler., Clausen, T., Laouiti, A., Qayyum, A., Viennot,L. (2001). Optimized link state routing protocol for ad hoc networks,*Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001),* 2001.

Ramkumar, M. (2009). On the Complexity of Probabilistic Key Predistribution Schemes, *to be presented in the Embedded Systems and Communications Security Workshop (ESCS 2009),* Niagara, NY, September 2009.

8. Appendix

8.1 A scalable key predistribution scheme

Unlike MLS, scalable KPSs are susceptible to collusions. For an (n,p)-secure KPS, an attacker with access to secrets of n nodes can compute a fraction p of all possible pairwise secrets. As long as p is low enough (say 2^{-64}) it is computationally infeasible for an attacker to even identity *which* pairwise secrets can be compromised by using the pool of secrets accumulated from n nodes.

8.2 A scalable key predistribution scheme

In the subset keys and identity tickets (SKIT) scheme (Ramkumar, 2009) defined two parameters m and M, the KDC chooses mM secrets, say, $K_{i,j}, 1 \leq i \leq m, 1 \leq j \leq M$ (which can be derived from a single master secret μ as $K_{i,j} = h(\mu, i, j)$).

The KDC chooses a public pseudo random function (PRF) $f()$ which generates a $m \log_2 M$ pseudo-random bits. For a node with identity A the output of the PRF $f(A)$ is interpreted as m $\log_2 M$-bits values, $a_i, 1 \leq i \leq m, 0 \leq a_i \leq M-1 \forall i$. Corresponding to the m indices, A is issued m secrets $K_{i,a_i}, 1 \leq i \leq m$. Node A is also issued mM identity tickets $I_{i,j} = h(K_{i,j}, A), 1 \leq i \leq m, 1 \leq j \leq M$. Identity tickets are conceptually similar to HMACs; however, while HMACs are not intended to be secrets, identity tickets provided to A are intended only for A.

Two nodes A and B can compute $2m$ common tickets. Computing any pairwise secret (say when A requires to compute K_{AB}) will require generating $m \log_2 M$ pseudo random bits to determine the indices of the m secrets assigned to B, followed by computation of m hashes. Every node requires storage for mM certificates.

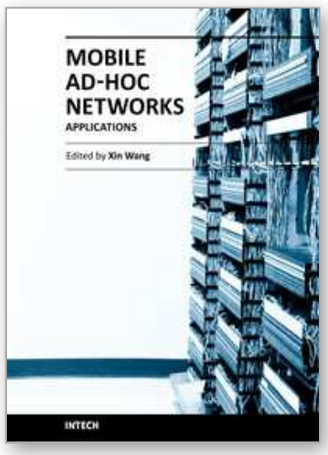
An attacker with access to secrets of n nodes $O_1 \cdots O_n$ can compute K_{AB} if the m secrets of each of the n nodes include $K_{i,a_i}, 1 \leq i \leq m$ *and* $K_{i,b_i}, 1 \leq i \leq m$. The probability of such an event is

$$p(n) \approx (1 - e^{-n/M})^{2m}. \tag{7}$$

For $m = 32$ and $M = 2^{16}$, $p(45{,}000) < 2^{-64}$, and $p(84400) \approx 2^{-30}$. For $m = 32$ and $M = 2^{16} \times 5$, $p(225{,}000) < 2^{-64}$, and $p(422{,}000) \approx 2^{-30}$.

If each node can afford 100 MB storage we can choose $m = 32$ and $M = 2^{16} \times 5$ to realize a scheme for which $p(225{,}000) \approx 2^{-64}$ and $p(422{,}000) \approx 2^{-30}$. Only the storage complexity is

increased. The computational overhead, which is influenced by the value $m = 32$ remains the same. Due to the low computational overheads, the computations can be easily performed inside the modest SIM cards to further alleviate the issue of exposure of secrets from a large number of nodes. An attacker desiring the exploit the collusion susceptibility of SKIT will have to successfully tamper with and expose secrets from several hundred thousand SIM cards.



Mobile Ad-Hoc Networks: Applications
Edited by Prof. Xin Wang

ISBN 978-953-307-416-0
Hard cover, 514 pages
Publisher InTech
Published online 30, January, 2011
Published in print edition January, 2011

Being infrastructure-less and without central administration control, wireless ad-hoc networking is playing a more and more important role in extending the coverage of traditional wireless infrastructure (cellular networks, wireless LAN, etc). This book includes state-of the-art techniques and solutions for wireless ad-hoc networks. It focuses on the following topics in ad-hoc networks: vehicular ad-hoc networks, security and caching, TCP in ad-hoc networks and emerging applications. It is targeted to provide network engineers and researchers with design guidelines for large scale wireless ad hoc networks.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Sivakumar Kulasekaran and Mahalingam Ramkumar (2011). APALLS: A Secure MANET Routing Protocol, Mobile Ad-Hoc Networks: Applications, Prof. Xin Wang (Ed.), ISBN: 978-953-307-416-0, InTech, Available from: http://www.intechopen.com/books/mobile-ad-hoc-networks-applications/apalls-a-secure-manet-routing-protocol

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen