Obtaining Non-repudiable Proof of Active Attacks in DSR

Kulasekaran A. Sivakumar, Mahalingam Ramkumar
Department of Computer Science and Engineering
Mississippi State University.

***Abstract*—Many secure MANET routing protocols have been proposed in the literature which employ non repudiable cryptographic authentication. However, they have not been explicitly designed to provide non repudiable proof of active attacks (NPAA). For example, Ariadne with digital signatures (ADS) is a secure extension of DSR which requires every node forwarding a route request to append a digital signature. Such signatures are carried forward all the way to the destination node to facilitate the destination to *detect* active attacks. However, ADS is not well-suited for providing NPAA due to the need for accused nodes to provide *affirmative defense*. We propose a secure DSR based protocol, APALLS, designed explicitly to provide NPAA. Apart from the eliminating the need for affirmative defense, another advantage of APALLS is that it does not require digital signatures to be carried over.**

I. Introduction

Routing protocols for mobile ad hoc networks (MANET) are rules to be followed by every node to co-operatively discover optimal paths and route packets between end-points (source and destination nodes). The presence of nodes that do not adhere to the rules, either deliberately, or due to malfunctioning, can have a deleterious effect on other nodes.

The types of attacks that can be inflicted by misbehaving nodes can be broadly classified into passive and active attacks. Passive attacks may involve non-participation or selective (selfish) participation. Active attacks involve illegal modifications to routing packets. Secure MANET routing protocols include explicit measures to address such attacks.

Cryptographic authentication is necessary to address both forms of attacks. Mechanisms for addressing passive attacks require monitoring of neighbors; for this purpose, a node processing a packet needs to know *who* sent that packet - which is made possible through a cryptographic authentication token accompanying the packet. Strategies for addressing active attacks typically require verification of multiple cryptographic tokens created by multiple of nodes.

Specific authentication strategies required to detect active attacks are however intricately tied to the nature of the underlying routing protocol. In this paper we restrict ourselves to the dynamic source routing (DSR) protocol. In DSR paths between a pair of nodes (source and destination) is established by flooding a route request (RREQ) packet originating from the source, in response to which the destination, or a node which is aware of a path to the destination, invokes a route-response (RREP) packet indicating the path between the end-points. The focus of this paper is on issues in obtaining non repudiable proof of active attacks on the dynamic source routing (DSR) protocol [1].

A. Contributions

Arguably, the most effective deterrent for active attacks is an ability to obtain non repudiable proof of active attacks (NPAA), which could lead to *revocation* of the attacker from the network. An obvious prerequisite for NPAA is the use of a non repudiable cryptographic authentication strategy, like digital signatures. This is not withstanding some unfortunate side effects of employing digital signatures, like substantial computational and bandwidth overhead for battery operated mobile devices, and increased susceptibility to simple denial of service (DoS) attacks. Specifically, compared to message authentication codes based on symmetric secrets, digital signatures impose 2 orders of magnitude higher computational burden. By sending random bits in place of signatures attackers can force receiving nodes to suffer the overhead for verifying the signature.

The primary contribution of this paper is an efficient protocol, APALLS - Ariadne with pairwise authentication and link-layer signatures - for NPAA in DSR. We argue that some of the pitfalls of Ariadne with digital signatures (ADS) include

1) the need for accused nodes to provide *affirmative defense*[1], due to which nodes could be accidentally penalized; and
2) the need for carrying over digital signatures (and public key certificates required to verify signatures) appended by every node in the path till the end of the path (the destination node).

In APALLS, *link-layer* (one-hop) signatures are not carried forward. Nevertheless, a signed packet from a node includes all the contextual information required to determine if a node did (or did not) engage in an active attack. Consequently, APALLS eliminates the need for accused nodes to provide additional affirmative defenses, while simultaneously reducing overhead.

The rest of this paper is organized as follows. In Section II is i) an overview of DSR, ii) the open-managed network model [7] for MANETs, and iii) key distribution schemes for facilitating cryptographic authentication.

In Section III we begin with an overview of Ariadne [2], a popular secure extension of DSR. In particular, we

[1]As we shall see later in this paper a falsely accused node will require to store and provide a per-hop hash reported by an upstream node to demonstrate that it did not engage in an active attack.

review Ariadne with digital signatures (ADS), and issues in obtaining NPAA using ADS. We argue that the main reason that ADS is ill-suited for providing NPAA is that even while a signed RREQ packet broadcast by a node C in a path $(\ldots, A, B, C, \ldots)$ includes the signatures of all upstream nodes $(\ldots, A, B)$, the signed packet from C still does *not* include all necessary information to determine if C had engaged in an active attack. This is rectified in the APALLS protocol proposed in this paper. Conclusions are offered in Section III-F.

II. Background

MANET routing protocols can be broadly classified into *proactive* protocols which strive to maintain consistent topology information of the subnet at all times, and *reactive* (or on-demand) protocols which attempt to discover routes only when necessary. In this paper we restrict ourselves to the popular on-demand dynamic source routing protocol (DSR) [1].

A. DSR

In DSR a node S desiring to find a path to a node T broadcasts a route-request (RREQ) packet $\mathcal{Q} = [S, q, T, n_h]$, indicating the source S, source sequence number q, destination T, and a hop-limit n_h. RREQ packets are flooded. In general, every node in the connected subnet will receive one RREQ from each neighbor, as each node will re-broadcast the RREQ once (usually the first RREQ received by the node). Every node rebroadcasting an RREQ inserts its identity.

For example, an RREQ initiated by S for a destination T, traversing through a path $(A, B, C, \ldots)$ is of the form $[(S, q, T, n_h), (A, B, C, \ldots)]$. When the RREQ packet reaches the destination[2] a route-response (RREP) packet is created by the destination, and relayed along the reverse path $(\ldots, C, B, A)$ to S. The source and destination may in general discover multiple paths as the destination T may receive one RREQ from every neighbor (of T).

1) Attacks on DSR: Attacks on DSR can be broadly classified into passive and active attacks. Passive attackers may perform eavesdropping on application data packets exchanged between nodes, or take selective part in the network. For example, a selfish node C may simply overhear all packets without announcing its presence (and thus not participate in routing) until it has the need to communicate with an other node, or over-hears an RREQ addressed to C.

Active attackers 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 (like source identity, hop count, sequence number, destination identity) and/or other upstream nodes. In this paper we restrict ourselves to addressing active attacks on DSR.

Several secure extensions of DSR [2] - [6] have been proposed in the literature which permit the source and/or destination to verify the cryptographic integrity of the path indicated in the RREP/RREQ, and thereby, detect inconsistencies in the advertised path. In [5] the authors argue that even strategies for detection in many popular protocols have severe issues in the face of collusion amongst attackers. On the other hand, in [6] it was argued that a perfect *detection* strategy has little practical utility as a mechanism for improving resiliency, and that strategies to *identify* malicious nodes responsible for the detected inconsistency, in order to keep them out of paths, lead to increased resilience.

B. Managed Open MANET Model

A MANET *network* is a collection nodes which agree on a network (routing) protocol. A subset of nodes belonging to a MANET network, which accidentally (or otherwise) find themselves in some geographical region, can come together to create a *MANET subnet*, and relay packets amongst each other. Thus, any node in a MANET subnet with access to a wide area network (like the Internet) can extend this service to all other nodes in the subnet. While a MANET network may include millions of nodes, most MANET subnets may not scale beyond a few hundred nodes.

The most important advantage of MANETs stems from their reduced dependence[3] on communication infrastructure. This feature makes them useful in scenarios where it may be impractical to set up expensive infrastructure, and in disaster scenarios involving failure of infrastructure.

The most common of MANET models is a managed-open model [7], where an *off-line* authority, which requires very little investment in infrastructure to manage the MANET network, performs the following basic functions:

1) *promulgating* the rules to be followed by every node;

2) *inducting* nodes into the network by acting as a key distribution center and providing nodes with secrets necessary to take part in the network; more specifically, such secrets permit inductees (nodes) to authenticate themselves to other nodes, by appending verifiable cryptographic authentication tokens like message authentication codes (MACs), or digital signatures; and

3) *ejecting* nodes from the network (revocation) - perhaps by circulating revocation lists.

In the rest of this section we outline a possible approach for an off-line trusted authority (TA) of a managed-open MANET to accomplish its tasks. This example is also used later as the basis for the scheme proposed in Section III.

1) Key Distribution for Induction: 1) The TA generates an asymmetric key pair (R_T, U_T), which is used to sign public keys of inductees;

2) the TA chooses a master secret K_μ which is used to bootstrap an efficient scheme [8] for establishing pairwise secrets between inductees; and chooses a strong preimage resistant pseudo-random function $h()$ (for example SHA-1).

[2]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.

[3]Unlike conventional networks which require an infrastructure of dedicated routers to route packets between hosts, in MANETs every node is simultaneously a network host and a router.

The TA is now ready to issue secrets to nodes. Every inductee is issued a unique identity. Without any loss of generality, we shall assume that the identity is a unique sequence number (incremented sequentially with every new node inducted into the network). A node assigned a sequence number q receives i) a secret $K_q = h(K_\mu, q)$; and ii) an asymmetric key pair (R_q, U_q), with a signed certificate C_q linking the public key with the sequence number q.

In addition, the TA provides a URL from where node q can download $q-1$ *public values* of the form

$$P_{qi} = h(K_q, i) \oplus h(K_i, q) \forall i < q \tag{1}$$

Two nodes X and Y compute a common secret

$$K_{XY} = \begin{cases} h(K_X, Y) & X > Y \\ h(K_Y, X) & Y > X \end{cases} \tag{2}$$

For example, if $X > Y$, node Y (which does not have access K_X) computes $K_{XY} = h(K_X, Y)$ as

$$\begin{aligned} K_{XY} &= h(K_Y, X) \oplus P_{XY} \\ &= h(K_Y, X) \oplus (h(K_X, Y) \oplus h(K_Y, X)) \end{aligned}$$

X can authenticate any message M to Y by appending a message authentication code (MAC) $m_{XY} = h(M, K_{XY})$ computed using the pairwise secret K_{XY}. In practice, the master secret K_μ could be a 512-bits. Secrets like K_A assigned to nodes, could be 160-bits long. The values like P_{AX} (and consequently, pairwise secrets like K_{AX}) could be 80-bit values, obtained by retaining only 80 LSBs of $h(K_A, X) \oplus h(K_X, A)$. The millionth inductee into the network will need about 10 MB of storage (for 1 million 10-byte public values) in the mobile computer. The ten millionth node will require 100 MB of storage.

To authenticate a message M such that any node can verify the authentication, the source X appends a digital signature

$$\Sigma_X = \langle M, R_X \rangle = f_{sign}(M, R_X). \tag{3}$$

The signature can be verified by any node which has a genuine copy of the public key U_X of X, as

$$f_{ver}(M, U_X, \Sigma_X) = TRUE. \tag{4}$$

The public key U_X, included in the certificate C_X issued by the TA, can be included along with the signature, or needs to be conveyed to all potential verifiers through other means.

2) Revocation: The TA periodically posts signed revocation lists, consisting of identities of nodes who have been revoked from the network. As in general nodes do not have access to the TA while they are operating in a MANET subnet, nodes may periodically visit a website operated by the TA to obtain the latest revocation list.

A revoked node is one that is ejected from the network before expiry of its subscription. This can happen if the TA has obtained incontrovertible proof of misbehavior of the node. For example, the contents of a signed packet transmitted by a node C at some t may have been provided to the TA by some node at some time $t_s > t$ (whenever the submitter has access to the TA). This packet may enable the TA to unambiguously determine if C did (or did not) perform an active attack. If C did engage in an active attack, the TA can add the identity C to subsequent revocation lists. The possibility of revocation can be an effective deterrent for nodes desiring to engage in such attacks.

III. Non-repudiable Proof of Active Attacks

In DSR active attacks can be performed by a node in the course of i) propagating an RREQ or ii) relaying an RREP or iii) relaying a route error (RERR) packet. In the course of propagating an RREQ, an active attack may take the form of i) inserting non existent nodes in the path; ii) modifying the values inserted by upstream nodes; and iii) deleting values inserted by upstream nodes. In this paper we shall restrict ourselves to active attacks on RREQ propagation as attackers have very little reason to engage in active attacks on RREP and RERR packets (as passively dropping RREP/RERR packets will have the same effect). Furthermore, that RREQ messages need to be modified at every hop (unlike RREP/RERR) renders providing NPAA of active attacks on RREQ more challenging.

A. Ariadne

Ariadne is a secure extension of DSR in which a secret shared between end-points is used to address node-deletion attacks through a per-hop hashing strategy. Ariadne suggests three different options to address node insertion attacks: i) Ariadne-TESLA, using TESLA [9]; ii) Ariadne-PA (APA) employing pairwise authentication facilitated by pairwise secrets between every pair of nodes; or iii) Ariadne-DS (ADS) using digital signatures. In this paper we shall restrict ourselves APA and ADS.

In all forms of Ariadne the RREQ relayed by a node includes an authentication token appended by the node. In APA the authentication token M_A appended by an intermediate node A is a MAC computed using the pairwise secret K_{AT} shared between A and the RREQ destination T. In ADS the authentication token is a digital signature Σ_A which can be verified by all neighbors of A, and the destination T. In addition, in all three flavors of Ariadne the RREQ transmitted by every node includes a per-hop hash value which is intended only for neighbors.

For an RREQ initiated by S with a sequence number q, destined for T over a path $A, B, C, \ldots$ the sequence of steps in ADS are as shown in the left column in Table I (the sequence of steps in APALLS are depicted in the right column of Table I).

ADS assumes that the source S and the destination T share a $\bar{K}_{ST}$. This secret is used to seed a "per-hop hashing" strategy with a value $\beta_S = h(\bar{K}_{ST}, \mathcal{Q})$, where $\mathcal{Q} = [S, S_q, T, n_h]$. Every intermediate node inserts its identity and a digital signature, which are carried forward all the way to the destination. In addition, the broadcast by every node includes the per-hop hash value: the value β_A from A is made available only to neighbors of A; the value β_B is made privy only to neighbors of B.

Every intermediate node forwarding the RREQ verifies the signature appended by the previous hop. Note that even though the RREQ received by C includes signatures of S, A, and B, node C *cannot* verify the signature Σ_A appended by A as Σ_A is computed over a value β_A which is not privy to C. Thus, only neighbors of node A (who have access to β_A), and the destination (who can compute β_A), can verify the digital signature.

When the destination receives the RREQ $\mathcal{Q}_q^E \parallel \beta_E$, say through a path (A, B, C, D, E), where

$$\begin{aligned}\mathcal{Q}_q^E &= (S \parallel q \parallel T \parallel n_h \parallel \Sigma_S) \parallel (A \parallel \Sigma_A) \parallel (B \parallel \Sigma_B) \parallel \\ &\quad (C \parallel \Sigma_C) \parallel (D \parallel \Sigma_D) \parallel (E \parallel \Sigma_E), \end{aligned} \tag{5}$$

the destination i) computes the per-hop hash seed β_S; ii) recursively computes $\beta_A \cdots \beta_E$, assuming that the path indicated is correct; iii) verifies that the computed β_E matches the value submitted by E; iv) using the computed values $\beta_A \cdots \beta_E$, verifies the signature appended by intermediate nodes (it is assumed that some additional mechanism exists to provide the destination with the public key certificates of all intermediate nodes).

If the RREQ passes all these tests, and if it is reasonable to assume that only entity X has access to the secrets of X, the verifiable signatures assures the destination of the authenticity of the nodes in the path. Note that the signature of any node (say) C demonstrates that C did indeed have access to β_B; this convinces the destination that C is a neighbor of B. Thus, it is not possible for an intermediate node to delete nodes (along with their signatures) before forwarding the RREQ. More specifically, C can delete B from the path only if it has access to the value[4] β_A (which is privy only to neighbors of A, and C is not one). The destination invokes an RREP which includes the entire path. Unlike the RREQ where every node inserts some fields, all fields of the RREP are immutable. The destination can simply sign the RREP indicating the entire path and relay the RREP over the reverse path (this is the reason we do not address active attacks on RREP in this paper).

B. Proof of Active Attacks

If any node introduces illegal modifications (that violate the prescribed protocol), we desire that a signed packet broadcast by the node will constitute incontrovertible proof of such illegal behavior. Specifically, when such a signed packet is provided to the TA, the TA should be able to determine that the node did engage in an active attack. More specifically, any signed packet broadcast by any node should permit the TA to unambiguously determine if the node did (or did not) engage in an active attack.

As a more concrete example, consider a node C in a path $(A, B, C, \ldots)$ between the source S and the destination T, where a node C receives $[\mathcal{Q}_q^B, \beta_B]$ (where $\mathcal{Q}_q^B = [\mathcal{Q}_q^A, (B, \Sigma_B)]$) from its upstream node B. In response assume that C broadcasts $[\mathcal{Q}_q^C, \beta_C]$ where

$$\mathcal{Q}_q^C = [\mathcal{Q}_q^{B'}, (C, \Sigma_C)] \tag{6}$$

Assume that the signed RREQ from C is made available to the TA.

Now C is said to have engaged in an active attack if

1) $\mathcal{Q}_q^{B'} \neq \mathcal{Q}_q^B$ (if any field inserted by any upstream node is modified by C), or
2) if $\beta_C \neq h(\beta_B \parallel C)$ (illegal choice of per-hop hash), or
3) the signature of the upstream node B (Σ_B) is inconsistent with $\mathcal{Q}_q^B$ and β_B (in which case C should have ignored the RREQ from B).

To determine if C had engaged in an active attack the TA requires access to the value β_B - both to verify that β_C is consistent with β_B, and to verify that Σ_B is consistent with $\mathcal{Q}_q^{B'}$.

It is important to note that it is possible for the TA to determine the per-hop values employed by intermediate nodes only *if* all nodes had acted in a consistent manner. However, if any node upstream of C (or C itself) had illegally modified the per-hop hash value, or introduced/deleted a node in the path, this synchronization is lost. As the TA cannot compute β_B from β_C (as the hash function $h()$ is pre-image resistant), the only practical option for the TA is to demand that C produce the value β_B that is consistent with the signature Σ_B in $\mathcal{Q}_q^{B'}$,

TABLE I
RREQ (FROM SOURCE S TO DESTINATION T, SEQUENCE NUMBER q) PROPAGATION IN ARIADNE (LEFT) AND APALLS (RIGHT) OVER A PATH $(A, B, C, D, \ldots)$.

$S:$	$\mathcal{Q}_q = [S, q, T, n_h]$	$\mathcal{Q}_q = [S, t, T, n_h]$
$S:$	$\beta_S = h(\mathcal{Q}_q^S, \bar{K}_{ST})$	$\beta_S = h(\mathcal{Q}_q, K_{ST})$
$S:$	$\Sigma_S = \langle \mathcal{Q}_q, \beta_S, R_S \rangle$	$\Sigma_S = \langle \mathcal{Q}_q, \beta_S, R_S \rangle$
$S:$	$\mathcal{Q}_q^S = [\mathcal{Q}_q, \Sigma_S]$	$\mathcal{Q}_q^S = [\mathcal{Q}_q]$
$S \rightarrow *$	$\mathcal{Q}_q^S, \{\beta_S\}$	$\mathcal{Q}_q^S, \{\beta_S, \Sigma_S, NULL, NULL\}$
$A:$	$\beta_A = h(\beta_S, A)$	$\beta_A = h(\beta_S, A)$
$A:$		$\nu_A^S = K_{AT}(\beta_S)$
$A:$	$\Sigma_A = \langle \mathcal{Q}_q^S, A, \beta_A, R_A \rangle$	$M_A = h(\mathcal{Q}_q^S, \nu_A^S, \beta_A, K_{AT})$
$A:$	$\mathcal{Q}_q^A = [\mathcal{Q}_q^S, (A, \Sigma_A)]$	$\mathcal{Q}_q^A = [\mathcal{Q}_q^S, (A, \nu_A^S, M_A)]$
$A:$		$\sigma_S = h(\Sigma_S), v_A = h(\sigma_S, NULL)$
$A:$		$\Sigma_A = \langle \mathcal{Q}_q^A, \beta_A, v_A, R_A \rangle$
$A \rightarrow *$	$\mathcal{Q}_q^A, \{\beta_A\}$	$\mathcal{Q}_q^A, \{\beta_A, \Sigma_A, \sigma_S, NULL\}$
$B:$	$\beta_B = h(\beta_A, B)$	$\beta_B = h(\beta_A, B)$
$B:$		$\nu_B^A = K_{BT}(\beta_A)$
$B:$	$\Sigma_B = \langle \mathcal{Q}_q^A, B, \beta_B, R_B \rangle$	$M_B = h(\mathcal{Q}_q^A, \nu_B^A, \beta_B, K_{BT})$
$B:$	$\mathcal{Q}_q^B = [\mathcal{Q}_q^A, (B, \Sigma_B)]$	$\mathcal{Q}_q^B = [\mathcal{Q}_q^A, (B, \nu_B^A, M_B)]$
$B:$		$\sigma_A = h(\Sigma_A), v_B = h(\sigma_A, v_A)$
$B:$		$\Sigma_B = \langle \mathcal{Q}_q^B, \beta_B, v_B, R_B \rangle$
$B \rightarrow *$	$\mathcal{Q}_q^B, \{\beta_B\}$	$\mathcal{Q}_q^B, \{\beta_B, \Sigma_B, \sigma_A, v_A\}$
$C:$	$\beta_C = h(\beta_B, C)$	$\beta_C = h(\beta_B, C)$
$C:$		$\nu_C^B = K_{CT}(\beta_B)$
$C:$	$\Sigma_C = \langle \mathcal{Q}_q^B, C, \beta_C, R_C \rangle$	$M_C = h(\mathcal{Q}_q^B, \nu_C^B, \beta_C, K_{CT})$
$C:$	$\mathcal{Q}_q^C = [\mathcal{Q}_q^B, (C, \Sigma_C)]$	$\mathcal{Q}_{(q,S)}^C = [\mathcal{Q}_q^B, (C, \nu_C^B, M_C)]$
$C:$		$\sigma_B = h(\Sigma_B), v_C = h(\sigma_B, v_B)$
$C:$		$\Sigma_C = \langle \mathcal{Q}_q^C, \beta_C, v_C, R_C \rangle$
$C \rightarrow *$	$\mathcal{Q}_q^C, \{\beta_C\}$	$\mathcal{Q}_q^C, \{\beta_C, \Sigma_C, \sigma_B, v_B\}$

[4]The only way a node who is not physically a neighbor of A can obtain the value A is by colluding with a neighbor of A. Ariadne was *not* designed to address attacks by colluding nodes.

and also satisfies $\beta_C = h(\beta_B, C)$ (β_C is included in the signed packet from C).

If every node stores the upstream per-hop hash received from the previous hop for every RREQ broadcast by the node in the past, we can expect C to produce this value. This is obviously an unrealistic expectation. Another option may be that when the destination detects an inconsistency, it sends a message to all nodes to store the upstream per-hop hash corresponding to the specific RREQ. However, if an innocent node D, which did receive a value β_C consistent with the signature Σ_C, had suffered a crash (and lost the value β_C), then D is likely to be construed as an active attacker due to its inability to defend itself by providing the value β_C. It is also possible that the destination T could collude with the attacker[5] C to frame D.

C. APALLS - Eliminating Affirmative Defense

The need for affirmative defense can be eliminated if the values inserted in the RREQ by every node includes an additional value - the per-hop hash submitted by its upstream node. As the packet sent by D also includes the per-hop hash submitted by C, D will not need to provide his value to the TA or the destination when challenged later. The TA can immediately verify that D did indeed act in a consistent manner.

However, for the security of the per-hop hash mechanism, this value needs to be protected from downstream nodes. This can be achieved if intermediate nodes share a secret with the destination, and this secret is used to encrypt the upstream per-hop hash inserted into the RREQ packet. Compared to ADS, the first modification in APALLS is the introduction of the encrypted-upstream per-hop hash as one of the fields inserted by every intermediate node. More specifically, if C receives the hash β_B from its upstream node, C, which shares secret K_{CT} with the destination T, inserts an additional value

$$\nu_C^B = K_{CT}[\beta_B] \tag{7}$$

in the RREQ, where the notation $Y = K[X]$ represents encryption of the value X using a key K using some standard block-cipher. We also use the notation $X = K^{-1}[Y]$ to denote decryption using the block-cipher.

1) One-hop Signatures: In ADS the signature appended by a node serve three purposes:

a) for verification by neighbors (an RREQ packet relayed by a node "claiming to be A" will not be honored by its neighbors unless the neighbors are able to verify the signature of A;
b) verification by destination to prevent node insertion attacks; and
c) non-repudiability (A cannot refute that a packet signed by A was not sent by A)

[5]In a path A, B, C, D, E between S and T, C engages in an active attack and the destination simply ignores it. After storing the value β_C for some duration, D may erase it as there is no apparent need for this value. Later, if T submits the packet to the TA, node D will not be able to provide the deleted value to defend itself.

If signatures are not carried forward, then the signature could potentially serve two of the three purposes (first and third). It is only for verification by the destination that we need to carry over all signatures in ADS. However, if every node shares a secret with the destination (which is necessary in any case to encrypt the upstream per-hop-hash) then intermediate nodes can simply append a MAC (as in Ariadne with pairwise authentication (APA)) instead of a signature. Thus in APALLS, as in APA, every intermediate node appends a MAC instead of a signature: the values inserted in the RREQ by a node C, consists of three values $C \parallel M_C \parallel \nu_C^B$. For purposes of non repudiation, and for one-hop authentication, intermediate nodes append a "link-layer" signature (which is not carried forward).

By signing the values included in the RREQ broadcast by C, viz., $\mathcal{Q}_C = \mathcal{Q}_q^B \parallel C \parallel M_C \parallel \nu_C^B$, and Σ_B, node C claims that i) the values $\mathcal{Q}_q^B$ and $\beta_B = K_{CT}^{-1}[\nu_C^B]$ was broadcast by B; and ii) B's signature was verified by C. If such a packet from C is submitted to the TA, the TA should be able to verify the veracity of this claim, by verifying B's signature.

Just as C's signature was computed over the signature of its previous hop B (to support C's claim that it did indeed verify B's signature), node B's signature will be computed over the signature of B's previous hop, A (or Σ_A is required to verify B's signature). Thus, to verify if C did (or did not) engage in an active attack, the TA needs the signatures of C, and its two previous hops, B and A. However, carrying over signatures to two hops is wasteful of bandwidth, especially since the nodes themselves can only verify the signature of the previous hop. APALLS addresses this issue by using a novel construct to carry over digests of signatures efficiently.

In APALLS the one-hop values that accompany C's include the per-hop hash β_C (as in ADS), and three additional values to facilitate NPAA:

i) the signature Σ_C;
ii) σ_B - a commitment for the signature of the upstream node B; and
iii) v_B a value required by the TA to verify the signature of the upstream node B.

Neighbors of C first compute $v_C = h(\sigma_B, v_B)$ to verify the signature of C. After verifying the signature, a downstream neighbor of C (say D) sends two additional values - the hash of the verified signature $\sigma_C = h(\Sigma_C)$, and the value v_C. Note that v_C is essentially a one-way function of the signatures of *all* nodes upstream of B; v_B is a one way function of all nodes upstream of A (which is a single node, S); v_A is NULL as there is no node upstream of its immediate upstream node S.

2) RREP and RERR: While RREQ packets modified at every hop by nodes that forward the packets, RREP and RERR packets are relayed unmodified from the source of the RREP/RERR packets. Such packets can be signed by the creator to ensure that any modification en route will be detected. As intermediate nodes do not need to modify the packets it is unnecessary to require the forwarding nodes to sign such packets.

D. Revocation Process

The process of revocation has three steps. The first is for some node to suspect that a packet sent by a node (a neighbor) may not be consistent. If D has reasons to believe that a packet sent by C was questionable, it simply stores the packet for submission to the TA at a later (convenient) time. The second step in this process is for the TA to verify if the signed packet by C is proof of an active attack. The final step is for the TA to add the offender to the next revocation list, and broadcast the list (for example, through a web-page).

1) Triggering Suspicions: The reasons for suspecting the packet from a neighbor could be many fold: some examples of when a node D observing a neighbor C could suspect C include:

- C's RREQ claims to have a node X upstream of C; and D has not had the opportunity to confirm that a neighbor X of C does exist;
- D overhears two RREQs with the same source / sequence number but with different immutable fields (like hop-count or destination)

Apart from suspicions resulting from monitoring, nodes may also be informed by the destination if an RREQ packet was detected to be inconsistent. When the destination T receives the an RREQ over a path (say, (A, B, C, D, E)) it computes $\beta_S = h(\mathcal{Q}_q, K_{ST})$ in the same manner computed by the RREQ source. The destination then proceeds to check the consistency of every node in the path. The values appended by an intermediate node C, viz., M_C and ν_C^B are deemed self-consistent by the destination T if

$$M_C = h(\mathcal{Q}_q^B, (C, \nu_C^B), h(K_{CT}^{-1}[\nu_C^B], C), K_{CT}). \tag{8}$$

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 its upstream node B is found to be self-consistent, or $M_B = h(\mathcal{Q}_q^A, (B, \nu_B^A), h(K_{BT}^{-1}[\nu_B^A], B), K_{BT})$. If any inconsistency is observed the destination informs all nodes to store the RREQ received from their upstream nodes (and submit the RREQs to the TA whenever it is possible to do so).

2) Verification by TA: When a signed RREQ from C is submitted to the TA, the TA takes the following steps:

① Verify that Σ_C is consistent with $\mathcal{Q}_q^C$, β_C, and $v_c = h(\sigma_B, v_B)$;

② "Obtain" $\Sigma_B{}'$ for the values $\mathcal{Q}_q^B$ and $\beta_B = K_{CT}^{-1}[\nu_C^B]$ and v_B (which according to C, were broadcast by B);

③ Verify if $h(\Sigma_S{}') = \sigma_B$. If so, C's claim is correct, and C is not an active attacker. If not C is an active attacker as it did not either verify B's signature, or illegally modified the value provided by B.

If the TA has access to the private keys of all nodes the TA can "obtain" $\Sigma_B{}'$ by simply computing $\Sigma_B{}'$. If private keys are *not* escrowed by the TA, the TA can (off-line) demand B to sign the values $\mathcal{Q}_q^B$ and $\beta_B = K_{CT}[\nu_C^B]$ and v_B. Thus, even in scenarios where the private keys are not escrowed by the TA, unlike ADS, nodes will only need access to their private key to avoid being penalized (revoked) accidentally. Any node which claims to not have access to its private key should be revoked in any case.

An advantage of escrowing private keys by the TA is that the verification of proof of attacks can be performed as soon as a packet is submitted to the TA. 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 to deliver "swift justice."

E. APALLS vs Other Ariadne Variants

As mentioned earlier in Section III-A Ariadne can employ TESLA [9] or pairwise secrets or digital signatures for intermediate node authentication. It was argued in [6] that Ariadne with pairwise secrets is substantially more efficient compared to Ariadne with TESLA. Specifically, [6] argued that the ability to establish private channels (as opposed to merely authenticated channels if TESLA is used) has many beneficial side-effects. While both Ariadne with TESLA and Ariadne with pairwise secrets do not facilitate *identification* of active attackers in the path, this can easily be rectified in the latter by mandating every node forwarding the RREQ to append an encrypted upstream per-hop hash value. This additional upstream per-hop hash has a different purpose in APALLS; as the upstream per-hop hash is included in the broadcast by a node, this eliminates the need to provide affirmative defense.

APALLS has many desirable features. Firstly, even colluding nodes which may include the source and/or destination cannot frame an innocent node. The RREQ sent by a node X (for a destination T) is a deterministic function of the signed RREQ received by X from its upstream node Y, and the pairwise secret K_{XT}. It is important to note that that even knowledge of the pairwise secret is not sufficient to produce a packet impersonating X as the private key R_X is required to compute the signature of X. The only way to force a node X to behave in an apparently inconsistent manner is to *modify* its symmetric secret K_X or public values like P_{XT} used for computing the pairwise secret K_{XT}. Nodes should ensure that their public values are write-proof.

Secondly, an RREQ sent by any node provides the entire contextual information required to determine if a node did act in a consistent manner. This eliminates the need to provide affirmative defense, and in turn ensures that good nodes will not be accidentally penalized.

That signatures are not carried over results in savings in a bandwidth overhead for signatures and public key certificates (certificates need to be broadcast only to neighbors). When carrying over signatures (as in ADS) we need to ensure that the signatures are short, for example, using elliptic curve based signatures instead of RSA. However the unfortunate side effect of such an approach is significantly increased verification complexity, and consequently, increased susceptibility to simple denial of service (DoS) attacks. If signatures are restricted only

to one hop we can afford to use longer length RSA signatures which require substantially lower verification complexity[6], and thus reduced susceptibility to simple denial of service attacks.

In summary, APALLS has the following properties:

- good nodes (which follow the protocol) will not be accidentally penalized;
- good nodes cannot be framed by another node, or even by a collusion of other nodes, and
- lower overhead as signatures and public key certificates are not carried forward.

F. Conclusions

We have outlined a secure DSR protocol, APALLS, which is an extension of Ariadne [2]. To the extent of our knowledge, APALLS is the only MANET protocol designed to cater for non repudiable proof of active attacks (NPAA).

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 protocols, if a node B announces a distance of 5 to a node S, the neighbor C downstream of B is expected to announce a distance 6. In a scenario where C advertises a distance 7, proving that C did (or did not) violate the protocol requires several other pieces *contextual* information like (for example) i) if B was indeed a neighbor of C at that time; ii) the distance 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 collision), etc..

While necessary, non repudiable authentication is not sufficient for providing NPAA. Specifically, while some ad hoc routing protocols like ARAN [7] and ADS employ non repudiable authentication, they do not address the issue of *how* a packet sent from a node can be used as a proof of an active attack.

APALLS eliminates two of the main shortcomings of ADS. Firstly, it removes the need to provide affirmative defense. Secondly it eliminates the need to carry over signatures by employing a novel construct where two values - a commitment to the signature of the previous hop, and a one-way function of all signatures of upstream nodes need to be provided only to neighboring nodes.

One of our current research focus is investigation of NPAA strategies for other MANET routing protocols like AODV, TORA, etc.

REFERENCES

[1] P. Johanson, D. Maltz, "Dynamic source routing in ad hoc wireless networks," Mobile Computing, Kluwer Publishing Company, 1996, ch. 5, pp. 153-181.

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

[3] J. Kim, G. Tsudik, "SRDP: Securing Route Discovery in DSR," IEEE Mobiquitous'05, July 2005.

[4] P Papadimitratos, Z. J.Haas, "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.

[5] G. Acs, L. Buttyan, and I. Vajda, "Provably secure on-demand source routing in mobile ad hoc networks," IEEE Transactions on Mobile Computing, vol. 5, no. 11, pp. 15331546, 2006.

[6] K.A. Sivakumar, M. Ramkumar, "Improving the Resilience of Ariadne," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

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

[8] M. Ramkumar, "On the Scalability of a "Nonscalable" Key Distribution Scheme," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

[9] A. Perrig, R. Canetti, D. Song, D. Tygar, "Efficient and Secure Source Authentication for Multicast," in Network and Distributed System Security Symposium, NDSS '01, Feb. 2001.

[6]By choosing smaller public exponents for RSA.