An Efficient Secure Route Discovery Protocol for DSR

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

***Abstract*— Ensuring cryptographic integrity of the route discovery process in on demand ad hoc routing approaches like DSR require the ability to verify that no nodes have been deleted from the path, and no node can be inserted in the path without a valid authentication. We discuss the need for *early detection of inconsistencies* involving inserted or deleted nodes in route request (RREQ) packets and investigate the challenges associated with catering for this requirement. We propose an efficient strategy to achieve this employing only symmetric cryptographic primitives, which is made possible due to a recently proposed multi-source broadcast encryption scheme. We outline a protocol for secure route discovery in DSR that employs such a security primitive, and provide quantitative estimates (through simulations) of gains that can be achieved by early detection of inconsistent RREQs.**

I. Introduction

Nodes forming multi-hop wireless mobile ad hoc networks (MANET) [1] are expected to *co-operate* to a very large extent to jointly construct routing tables and deliver packets to one or more destination nodes which may be many hops away from the source. Efficient solutions to the problem of routing in MANETs can be challenging under the presence of malicious nodes that could deliberately violate the protocol and / or propagate misleading information. Secure routing protocols usually mandate cryptographic authentication *to reduce the degrees of freedom* of attackers to violate rules.

In this paper we restrict ourselves to the problem of securing on demand source routing based protocols. Many secure routing protocols based on the dynamic source routing (DSR) [2] protocol have been proposed in the literature. In such DSR-like protocols, a source node desiring to find a path to a destination floods a route request (RREQ) packet. Each node forwarding the packet inserts its ID. The destination sends a route response (RREP) along the reverse path when a RREQ packet reaches the destination.

Secure DSR protocols [3] - [5] employ cryptographic authentication to facilitate verification the integrity of the established route. However the nature of the protocol, and the specific cryptographic primitives used for authentication, will play a large role in determining *when* and by *whom* inconsistencies can be detected. In most secure DSR protocols, malicious modifications to RREQ packets *cannot* be detected by intermediate nodes that forward the RREQ. In some protocols the destination can detect inconsistencies and drop such requests. In some only the source, at the end of the reverse path, can detect inconsistencies after the RREP reaches the source.

Obviously, it would be very desirable to facilitate intermediate nodes to be able to detect inconsistencies in RREQ to avoid onwards relay of defective RREQs, which after wasting network bandwidth, will ultimately fail. Furthermore, inhibiting such RREQs will also facilitate discovery of other paths which would otherwise not have been detected (as they may be preempted by the bad RREQs). In this paper we discuss some of the issues that render early detection difficult, and propose an efficient solution, employing only symmetric cryptographic primitives, for this purpose.

In Section II of this paper we provide an overview of DSR and secure DSR extensions. We provide a generalized model of secure DSR protocols, and discuss the some of the issues that render early detection difficult. In Section III of the paper provides an overview of a recently proposed [7] multi-source broadcast encryption scheme and its utility in facilitating two-hop authentication. Section IV outlines a secure route discovery protocol (SRD) for source routing which employs the proposed two-hop authentication strategy to facilitate early detection of inconsistencies in RREQ packets. Section IV also includes simulation results to provide quantitative estimates of the advantages realized by early detection. Conclusions are offered in Section V.

II. Secure DSR Protocols

In DSR the route discovery process starts by broadcasting of a route request (RREQ) packet by the source, indicating the source, the destination, a unique sequence number and a hop-limit. Such RREQ broadcasts are flooded. The sequence number and hop-limit keep the flooding in check. Every node will forward only one RREQ packet with the same (source, sequence number) pair.

Each node relaying the RREQ packet appends its ID / network address to it. When the RREQ packet reaches the destination the node sends a route response (RREP) packet along the reverse path (the path through which it received the RREQ) - as each hop is explicitly indicated in the RREQ.

A. Authentication Immutable and Mutable Fields

In secure DSR protocols the RREQ packets that are forwarded consists of immutable and mutable fields. We shall henceforth represent the immutable fields of an RREQ by $rreq$. The immutable field specifies the source, destination, sequence number, maximum hop counts, and also typically

1930-529X/07/$25.00 © 2007 IEEE

includes an authentication introduced by the source (for example a digital signature).

The mutable fields, as the name indicates, are modified by every intermediate node. Typically every intermediate node may insert some form of authentication to validate the alterations made to the RREQ. In secure DSR extensions the mutable fields include the path, and authentication information appended by intermediate nodes.

For example in the case of an RREQ from S to D passing through a path $(A, B, C, \ldots,)$, a typical structure of the RREQs relayed by different nodes in the path may be as follows

$$\begin{array}{ll} S \rightarrow * & \text{RREQ}_0 = [\text{rreq}_0 \parallel S_S] \\ A \rightarrow * & \text{RREQ}_1 = [\text{RREQ}_0, (A), (S_A)] \\ B \rightarrow * & \text{RREQ}_2 = [\text{RREQ}_1, (A, B), (S_A, S_B)] \\ C \rightarrow * & \text{RREQ}_3 = [\text{RREQ}_2, (A, B, C), (S_A, S_B, S_C)] \end{array} \quad (1)$$

The specific nature of the authentication inserted by any node will determine *who* can verify the introduced authentication, and *when* it can be verified. If digital signatures are used by an intermediate node A, any one with access to the certified public keys of A can verify the authenticity of the signature. If the authentication introduced by A is a hashed message authentication code (HMAC) based on some secret K, only entities that share the secret K can verify the introduced authentication. For example if every pair of nodes shares a (pairwise) secret, then authentication inserted by A could be based on the secret K_{AD} it shares with the destination D. In Ariadne [4] which employs a time-sensitive authentication strategy relying on one-way hash chains, only the source of the RREQ, at the end of the reverse path, can verify the HMACs inserted by intermediate nodes.

1) Carrying Over Authentication: If digital signatures are used for authentication, the authentication introduced by node A for instance, can be verified by all nodes downstream of A. For example in a path (A, B, C, E, F) from S to D, if a malicious node C modifies any of the mutable fields introduced by B or A, nodes downstream of C can verify that such modifications are not consistent with the signatures of node B or A. However, mandating that every node insert a signature (and perhaps a certificate, if certificates cannot be distributed offline) before forwarding an RREQ implies very large bandwidth overheads for the RREQ, in addition to the computational overheads imposed by requiring every node to check the signatures of all upstream nodes.

One reasonable trade-off is to carry over the appended authentication only for two-hops. For instance, the authentication introduced by A could be verified by B and C and stripped off by C. Similarly while C forwards the authentication inserted by B onwards, this is stripped by downstream neighbors of C like E. In such a scenario where authentication is carried over only for two hops, note that colluding nodes could perpetuate misrepresentations. For example node B could make some illegal changes to the RREQ sent by A, which will be ignored by C (as C colludes with B).

B. Node Deletion Attacks

While verification of appended authentication by downstream nodes can prevent nodes from illegally inserting fictitious nodes in the path or modifying the mutable fields appended by nodes upstream, a simple attack for an attacker is to just remove an immediate upstream node (or a set of upstream neighbors) from the path. For example, if node C removes all fields inserted by B (both the ID B from the path, and the authentication appended by B), in effect node C claims to have received the RREQ directly from A. Even if the appended authentication is carried over all the way to the destination there is nothing inconsistent that becomes apparent from verification of the authentication appended by intermediate nodes. Similarly, C can also remove both[1] B and A from the path (along with the appended authentication).

Hu et al [4] proposed an elegant per-hop hashing strategy to overcome such deletion attacks. In such a strategy, the source of the RREQ sends an additional value $\beta_0 = h(\text{rreq}, K_{SD},$ where K_{SD} is a secret shared by the source and the destination. The node A replaces β_0 with $\beta_1 = h(\beta_0, A)$ before it forwards the RREQ onward. Similarly node B replaces β_1 with $\beta_2 = h(\beta_1, B)$, and so on. In order to ensure that intermediate nodes do not change the per-hop hash value, such values are also authenticated. For example, the authentication S_A introduced by A is for the quantity $[\text{rreq} \parallel (A) \parallel \beta_1]$. Similarly S_B, the authentication introduced by B, is for the quantity $[\text{rreq} \parallel (A, B) \parallel (S_A)\beta_2]$. When the destination receives the RREQ with some value "$beta_i$" and i nodes in the path, the destination can verify that β_i is consistent with all nodes in the path. Note that this is possible because the source and destination both share a secret K_{SD} and can thus evaluate $\beta_0 = h(\text{rreq}, K_{SD}$.

With the per hop hashing strategy, C cannot remove B from the path. By removing B from the path, C is implicitly claiming that it is a one hop neighbor of A. However, to prove that it is indeed a one hop neighbor of A, C needs to have access to the value β_1 send by A (which C does not have access to as only true neighbors of A are privy to this value).

Obviously any two nodes colluding together can delete all nodes in the path between them. Thus far there is is simply no solution that caters for assuring integrity of the path in the face of colluding nodes (and this paper does *does not* claim to provide a solution to this problem). Thus most secure routing protocols strive to assure integrity of the path discovery process only under the face of non colluding nodes. Under these circumstances, carrying over authentication by more than 2 hops to prevent illegal node insertions by colluding nodes is perhaps not so useful.

1) Per-hop Hashing and Carrying Over Authentication: One of the unfortunate side effects of the per-hop hashing scheme is that it even renders carrying over authentication to two-hops (which is required to prevent insertion attacks even by non-colluding nodes) impossible. For instance in a

[1]However C cannot afford to remove A and leave B in the path as the authentication appended by B will not be consistent without A in the path.

1930-529X/07/$25.00 © 2007 IEEE

path $(A, B, C, \ldots,)$ between S and D, node C can no longer verify the authentication appended by A. Specifically, the modifications introduced by A include a quantity β_1 which only one-hop neighbors of A should be privy to. Thus if A's authentication includes β_1, C cannot verify the appended authentication. On the other hand, if the authentication appended by A does not include β_1, the intermediate node B can modify β_1. Thus only neighbors of A, and the destination (which can calculate any β_i as it has access to β_0) can verify the authentication appended by A.

2) Early Detection of RREQ Inconsistencies: Thus the use of per-hop hashing technique is thus not conducive to early detection of RREQ failures. In order to facilitate identification of inconsistent RREQs by intermediate nodes, we need some strategy that does not rely on per hop hashing, but is still able to prevent insertion attacks (assuming no two nodes collude together).

To see how this can be done, consider the scenario where a RREQ travels along a path $(A, B, C, E, \ldots,)$, and node C removes B from the path and announces a path (A, C). What we desire now is for downstream neighbors of C (receiving such an RREQ from C) to recognize that A cannot possibly be a neighbor of C. If this is possible, the bad RREQ will be dropped as desired.

More specifically, for a RREQ received through a path $\cdots B \rightarrow C \rightarrow E$, node E should be able to verify

1) the authentication appended by B and C
2) that C is a neighbor of E, and
3) that B is a neighbor of C (to prevent node deletion attacks)

One way of realizing the above requirements is to ensure that every node has complete knowledge of their two-hop topology. However a node cannot merely afford to trust their one-hop neighbors to provide them with a list of *their* neighbors, as a malicious C could easily claim that A is also a neighbor. Thus nodes require "authenticated knowledge" of the two-hop topology. This can be achieved if the "neighbor list" information supplied by all neighbors of a node (from which two-hop nodes can be detected), also includes the authentication appended by every node in the list. Obviously, this is an expensive proposition, especially in scenarios involving highly dynamic nodes.

In the next section we propose an efficient strategy for realizing this requirement.

III. Efficient Two-Hop Authentication

The two-hop authentication strategy proposed in this section requires nodes to only maintain a consistent one-hop topology. This is made feasible by the use of a recently proposed *multi-source broadcast encryption* (MSBE) scheme [7], in conjunction with maintenance of a secure reliable delivery neighborhood (RDN) by every node.

We shall first discuss why maintaining a secure RDN is necessary even if we do not require two-hop authentication. We shall then discuss the implementation of two-hop authentication employing MSBE.

A. Secure RDN

Most secure on-demand routing protocols incorrectly make an assumption that all links are bidirectional. As a justification for this assumption it is often pointed out [3], [5] that MACA[2] protocols like 802.11 employ RTS / CTS handshakes which rule out use of one-way links. However RTS / CTS handshakes can only be used for *unicast* packets like RREP where the sender explicitly specifies an intended receiver. Such handshakes cannot be used for RREQ packets that are flooded. Thus an RREQ packet transmitted by a node can reach a "neighbor" who does not have a reverse link.

Furthermore, even if RREQ packets are conveyed by individually unicasting them to every neighbor, it still does not prevent a "neighboring node" without a return link from gaining access to the packet. If such nodes forward the RREQ, the RREPs invoked in response to such RREQs will fail.

The assumption that "one-way links do not exist" has to be supported by some proactive means to ensure that such links cannot be *used*. One way of realizing this is for every node to proactively identify nodes within their RDN and supply such nodes with a secret. Thus a node A provides a secret K_A to all nodes in its RDN. All transmissions by A could be encrypted with the secret K_A to ensure that "neighbors" not in the RDN cannot gain access to transmissions from A.

B. Multi-Source Broadcast Encryption

Broadcast encryption (BE) ([8]) provides a means of establishing a shared secret between g privileged nodes, out of a universe of N nodes, where $g + r = N$, and the r nodes which are *not* provided with the secret are usually referred to a "revoked" nodes. For BE applications typically $r << N$.

Most popular BE schemes in the literature cater only for encrypted broadcasts by a single source. However such schemes can be extended to support BE by any node by using asymmetric cryptographic primitives where the encryption keys are public and private decryption keys are available to all receivers. Recently however, a family of MSBE schemes have been proposed that do not require asymmetric primitives. In such schemes the encryption secrets and decryption secrets are related through a simple one-way function. We shall see that the MSBE scheme is very well suited for achieving efficient two-hop authentication.

In the MSBE scheme in [7], a key distribution center (KDC) chooses k secrets $K_1 \cdots K_k$, a simple one way function $F()$, and a cryptographic hash function $h()$. The public function $F(A) = \{A_1, A_2, \cdots A_m\}$ determines the *indices* of secrets assigned to node A. Now node A is assigned m *decryption* secrets $\mathbb{S}_A$, and additionally, k *encryption* secrets $\mathfrak{S}_A$, where

$$\begin{aligned} \mathbb{S}_A &= \{K_{A_1}, K_{A_2}, \ldots, K_{A_m}\} \\ \mathfrak{S}_A &= \{K_j^A = h(K_j \parallel A)\}, 1 \leq j \leq k \end{aligned} \quad (2)$$

Let $\mathbb{U}$ represent the set of all nodes that have been provided

[2]Medium access collision avoidance or MACAW - MACA for Wireless.

1930-529X/07/$25.00 © 2007 IEEE

with encryption[3] and decryption secrets, and let $\mathbb{N}_A \in \mathbb{U}$ be a small subset of r nodes. The node A can now convey a broadcast secret T_A to all nodes *except* the r "revoked" nodes in the set $\mathbb{N}_A$, by encrypting the broadcast secret T_A with a subset of its encryption secrets $\mathfrak{S}'_A \in \mathfrak{S}_A$. The specific indexes of the encryption keys chosen (as part of $\mathfrak{S}'_A$) for this purpose are determined uniquely (through a public algorithm) which guarantees that

1) none of the r revoked nodes will have access to *any* of the keys in $\mathfrak{S}'_A$, and
2) Any other node (in $\mathbb{U} \setminus \mathbb{N}_A$) will have access to at least one of the secrets in $\mathfrak{S}'_A$ with a high probability.

To convey the secret to nodes in the set $\mathbb{U} \setminus \mathbb{N}_A$ node A constructs a broadcast message

$$\begin{aligned} \mathbb{B}_A &= [\mathbb{N}_A \parallel \{\mathfrak{S}'_A(K_A)\} \parallel M_{T_A}], \quad (3) \\ M_{T_A} &= h(\mathbb{N}_A, \mathfrak{S}'_A(K_A), T_A) \end{aligned}$$

For example, assume that the list of nodes to be revoked by A are X and Y. Let us assume that the indices of *encryption* secrets chosen by A for this purpose are $4, 21$ and 46. The secrets are chosen is such a way that none of the revoked nodes have *decryption* secrets with indices $4, 21, 46$. Now

$$\begin{aligned} \mathbb{N}_A &= \{X, Y\}, \quad (4) \\ \mathfrak{S}'_A(K_A) &= \{K_4^A(T_A), K_{21}^A(T_A), K_{46}^A(T_A)\} \quad (5) \end{aligned}$$

and M_{T_A} is a HMAC based on the secret T_A.

C. Efficient Two-hop Authentication

Apart from maintaining a secure RDN every node also constructs a broadcast encryption message which *explicitly* revokes all nodes in its RDN. Thus a node A with neighbors B, G, H in its RDN will construct a broadcast encryption message $\mathbb{B}_A$ that revokes nodes B, G and H (or $\mathbb{N}_A = \{B, G, H\}$) and contains encrypted versions of a secret T_A protected from the nodes in the RDN of A. In other words

1) K_A is a secret provided to all nodes in the RDN of A.
2) T_A is a secret protected from all nodes in the RDN of A.

Whenever the RDN of A changes, a new $\mathbb{N}_A$ is constructed by A. Whenever A relays or initiates an RREQ for the first time (since its RDN changed) it encloses the broadcast message $\mathbb{B}_A$ along with the RREQ. All one-hop neighbors of A cache this message till it is overwritten by another such message sent by A (possibly after a change in the RDN of A).

Any node forwarding an RREQ also includes a HMAC based on the broadcast secret. For example, in a scenario where an RREQ reaches D through the path $\cdots A \rightarrow B \rightarrow C \rightarrow D$, D can verify the authentication appended by B. To provide D with access to T_B, along with the RREQ, C also forwards the BE message $\mathbb{B}_B$ of its predecessor B.

The fact that node C is explicitly revoked in the BE message of B indicates that C is a neighbor of B. For a malicious C which desires to delete node B from the path, C has to produce a BE message from A which revokes C (to convince downstream nodes that A is a neighbor of C to advertise the path $A \rightarrow C$). C cannot forge a BE message from A which includes C as a revoked node. Every secret used in such a forged BE message corresponds to secrets C does *not* possess.

Note that A has no authenticated knowledge of its two hop topology. Even though A has access to B's BE message which indicates that C is a neighbor, this claim is *not* verified by A. This claim will be verified only if A has the opportunity to forward an RREQ that reaches A through the path $C \rightarrow B \rightarrow A$. In other words two-hops secrets are established and two hop neighbors identified only when necessary.

Furthermore, it does not matter if a node Y currently two-hops away from B (and had gained knowledge of T_B) moves within one-hop of B. When B accepts a Y into its RDN, it changes the RDN secret K_B and the two-hop secret T_B. Similarly it does not matter if a node 2 hops away suddenly powers itself off of even moves to a 3 hop distance. In other words, all that a node has to do is to maintain a one-hop topology.

For scenarios where each node has 5 neighbors on an average and say 25 nodes within the two-hop radius, a BE message will need to include 5 to 10 encryptions of the broadcast secret. Thus a BE message will require bandwidth overheads comparable to that of digital signatures, but very low computational overheads. However the BE message does not need to be forwarded with every RREQ. A node B broadcasts this meesage to its neighbors only when its RDN changes. Furthermore, a neighbor C (of B) does not have to forward the BE message from B every time it forwards an RREQ from B. Node C forwards the BE message from B only when ① it forwards an RREQ with B as the predecessor, *and* ② there is a change in the either the RDN of B or the RDN of C. More specifically, a change in the RDN of B invalidates the old BE message from B. A change in the RDN of C may imply that new nodes may be present in its RDN who have not received B's BE message the last time it was forwarded by C.

IV. A Secure Route Discovery Protocol

The secure route discovery (SRD) protocol to be outlined in this section employs broadcast encryption for two-hop authentication. The protocol assumes the presence of an *off-line* KDC who has distributed encryption and decryption secrets for the MSBE scheme. Further, the SRD protocol also assumes the existence of a suitable KDS for pairwise authentication of nodes. Recently schemes that are well suited for this purpose have been identified [10]. Specifically, the scheme in [10] takes advantage of the fact that even mobile devices can have easy access to large amounts of inexpensive storage (for example pluggable flash cards). Even with 100 MB of storage[4] of public values (that need not be protected) per device,

[3] All nodes that need to perform encrypted broadcasts require encryption keys. For our purposes all nodes are equipped with both encryption and decryption keys.

[4] With SD cards supporting several GBs of storage already being common, this is perhaps a reasonable requirement.

1930-529X/07/$25.00 © 2007 IEEE

the key pre-distribution scheme in [10] can resist collusions of over 100,000 nodes pooling their secrets together. More importantly, it mandates *very low* computational complexity (a *few tens* of symmetric cipher operations).

Every node maintains a secure RDN by providing a group secret to every node in the RDN (by encrypting the group secret individually using pair-wise secrets shared with neighbors). We shall represent the one-hop RDN secret of a node A (provided to all nodes in the set $\mathbb{N}_A$, the RDN of A) by K_A, and the broadcast secret (which is *protected* from all nodes in the set $\mathbb{N}_A$) by T_A.

A. RREQ Propagation

Let us now consider a scenario where a source S desires to find a path to a destination D. The source creates a RREQ packet with immutable fields

$$rreq = [S \parallel D \parallel seq \parallel h_c] \tag{6}$$

where seq is a sequence number and h_c is the maximum hop count. The node S now broadcasts an RREQ packet

$$\begin{array}{rcl} RREQ_0 & = & [S \parallel K_S([rreq \parallel M_0 \parallel h_0])] \\ h_0 & = & h(rreq, K_{SD}) \\ M_0 & = & h(rreq, h_1, T_S), \text{ where } h_1 = h(h_0) \end{array} \tag{7}$$

where K_{SD} is a secret shared between S and D. In other words, h_0 is a HMAC meant for verification by the destination, and M_0 is HMAC for verification by two-hop nodes. Note that all fields of the RREQ packet are encrypted by the one-hop secret of S.

A node A one hop from S decrypts the RREQ packet and broadcasts

$$\begin{array}{rcl} RREQ_1 & = & [A \parallel K_A([rreq \parallel (A) \parallel M_0 \parallel M_1 \parallel h_1])] \\ M_1 & = & h(rreq, (A), h_2, T_A), \text{ where } h_2 = h(h_1) \end{array} \tag{8}$$

A node B one-hop away from A and two-hops away from S decrypts the RREQ, verifies that h_1 sent by A is consistent with the HMAC M_0 appended by S. Having verified M_0, node B strips off M_0 and appends an HMAC M_2 for verification by nodes two hops downstream of B. Thus B airs

$$\begin{array}{rcl} RREQ_2 & = & [B \parallel K_B([rreq \parallel (A,B) \parallel M_1 \parallel M_2 \parallel h_2])] \\ M_2 & = & h(rreq, (A,B), h_3, T_B), \text{ where } h_3 = h(h_2) \end{array}$$

1) RREP: When the node D receives the RREQ packet with a per-hop hash value h_i and i nodes in the path, D can verify that h_i is consistent with h_0 (which D can evaluate as it is based on a secret K_{SD} shared between the source and destination). Let us assume that the RREP reached the destination through a path $(A, B, \ldots, W, X, Y, Z)$

The RREP raised by D takes the form

$$rrep = [S \parallel D \parallel seq \parallel (A, B, \ldots, W, X, Y, Z)].$$

Now D unicasts the RREP packet

$$\begin{array}{rcl} RREP_0 & = & [D \parallel K_D([rrep \parallel q_0 \parallel M_{DY}]) \\ M_{DY} & = & h(rrep, q_1, K_{DY}), \text{ where } q_1 = h(q_0) \end{array}$$

Note that the RREP packet has two HMACs - q_0 for verification by the source at the end of the RREP, and M_{DY} for verification by a node Y two hops away in the RREP path.

The RREP packets relayed by nodes Z and Y take the form

$$\begin{array}{rcl} RREP_1 & = & [Z \parallel K_Z([rrep \parallel q_1 \parallel M_{DY} \parallel M_{ZX}]) \\ M_{ZX} & = & h(rrep, q_2, K_{ZX}), \text{ where } q_2 = h(q_1) \\ RREP_2 & = & [Y \parallel K_Y([rrep \parallel q_2 \parallel M_{ZX} \parallel M_{YW}]) \\ M_{YW} & = & h(rrep, q_3, K_{YW}), \text{ where } q_3 = h(q_2) \end{array}$$

2) Comparison With Other Secure DSR Protocols: In the secure routing protocol (SRP) [3] by Papadimitritos et al., intermediate nodes do not introduce any authentication. Thus even external nodes can take part and disrupt the routing process. In Ariadne [4] every intermediate node appends a HMAC based on a TESLA key that will not be disclosed at least until the destination receives the RREQ. The TESLA keys used for authentication during the forward path are released during the reverse path. Thus at the end of the RREP the destination can discover node deletion attacks. Node insertion attacks can be discovered by the source at the end of the reverse path. If Ariadne is used in conjunction with pairwise secrets instead of TESLA intermediate nodes append a HMAC based on the pairwise secret they share with the destination. In this case the destination can detect both insertion and deletion attacks.

In [5] the authors present many different forms of authentication strategies for securing route discovery. The main focus of the protocols in [5] is to reduce the overheads for carrying over authentication by employing authentication strategies that can be aggregated to save bandwidth. However the schemes proposed in [5] do not consider node deletion attacks.

B. Need for Early Detection

One of the primary motivations for SRD is to ensure that malicious modifications to RREQ are detected early so that defective RREQs can be dropped as soon as possible. To see how dropping malicious RREQs soon enough can improve the performance of the route discovery process consider the topology in Figure 1 where D receives two RREQs, through paths (A, B, C, E) and (J, K, L, M, N, P, Q). Assume that both paths have a malicious node - say C in the first path and N in the second path - which perform illegal modifications to the RREQ which are recognized by the destination D.

The fact that no good path was discovered in this instance, does *not* necessarily mean that no good path *exists*. In this specific scenario, several good paths (without attackers) like (J, K, L, G, H, U, E, F), (A, B, G, H, U, E, F), (J, K, L, M, U, E, F), (J, K, L, M, U, V, F), (J, K, L, M, U, V, P, Q) exist. Unfortunately, as the RREQ relayed by C reaches node E earlier than RREQs from other paths, the first three paths will not be discovered. Similarly as node U receives the RREQ from C first, the fourth and fifth good paths will not be discovered. On the other hand, if defective RREQs can be detected immediately and stopped from further

1930-529X/07/$25.00 © 2007 IEEE

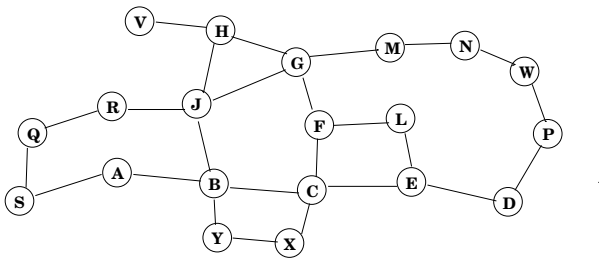

Fig. 1. Topology of an ad hoc subnet for illustrating the advantages of early detection of inconsistencies in RREQs.

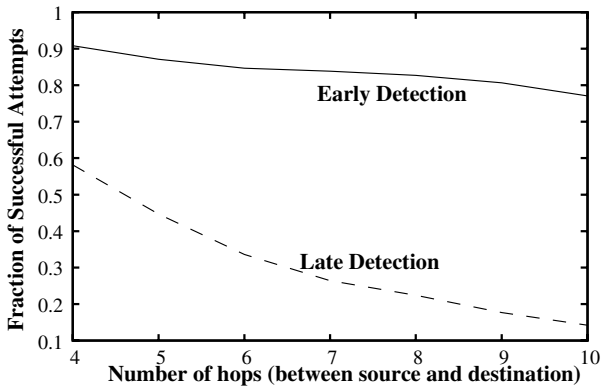

Fig. 2. Plots depicting the quantitative improvement in performance realized due to early detection of inconsistencies in RREQ packets.

propagation, the chances of discovering good paths can increase significantly.

1) Simulations: To obtain a more quantitative estimate of the benefits of early detection of inconsistencies in RREQs we have performed extensive simulations to determine the percentage of route discovery attempts (between randomly chosen node pairs) that succeed.

For the simulations we generated many random realizations of 200 nodes in a square with unit edges. The range of each node was assumed to be 0.1 units. Out of the 200 nodes 40 nodes were assigned as malicious. It is assumed that the malicious node will illegally modify the RREQ. In our simulations each node had (on an average) 5 neighbors. Note that each node could receive as many RREQs as the number of its neighbors. We assume that the route discovery attempt between two nodes fail if *every* such RREQ path includes a malicious node.

We simulated RREQ propagation between every pair of good nodes. The simulation results are shown for two cases ① bad RREQs are detected only by the destination (late detection) ② bad RREQs are detected within two hops (early detection) and stopped from further propagation. Simulation of RREQ propagation was performed for over 200,000 node pairs, chosen from 5 different random realizations of the network. In each realization 40 nodes were randomly assigned to be malicious. For purposes of comparison between the two cases under different hop counts between the source and the destination, the plots have the hop-count between the chosen source-destination pair (in terms of the number of hops in the shortest path) as the x-axis. The y-axis is the fraction of node-pairs that succeed in the route discovery attempts.

The dashed line represents late detection and the solid line represents early detection. As seen from the plots, early detection of RREQ inconsistencies can substantially improve the performance of any on demand routing algorithm by preventing preemption of good paths by defective RREQs.

V. Conclusions

We discussed the need for early detection of inconsistencies in RREQ packets and proposed an efficient strategy employing only symmetric cryptographic primitives to achieve this requirement. This was achieved by mandating every node to maintain a consistent one-hop RDN information and providing 1) a secret to all nodes in the RDN and 2) a broadcast encryption message that carries a secret accessible by any node *except* the nodes in the RDN. We pointed out the existence of efficient KDSs for pairwise authentication to facilitate establishment of RDN secrets and an efficient multi-source broadcast encryption scheme to broadcast a secret to any node except the nodes in the RDN. We outlined a secure route discovery protocol (SRD) for DSR and obtained quantitative estimates of the advantages of facilitating early detection of inconsistencies like node deletions and insertions in RREQ packets.

References

[1] Web Link, http://www.ietf.org/html.charters/manet-charter.html
[2] D. Johnson, D. Maltz, Y-C. Hu, J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks," Internet Draft, draft-ietf-manet-dsr-05.txt, June 2001.
[3] 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.
[4] Y-C Hu ,A Perrig,. D B.Johnson, "Ariadne:A Secure On-Demand Routing Protocol for Ad Hoc Networks," The 8th ACM International Conference on Mobile Computing and Networking, Atlanta, Georgia, September 2002.
[5] J. Kim, G. Tsudik, "SRDP: Securing Route Discovery in DSR," IEEE Mobiquitous'05, July 2005.
[6] 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.
[7] M. Ramkumar, "Broadcast Encryption with Probabilistic Key Distribution and Applications," *Journal of Computers*, June 2006.
[8] A. Fiat, M. Noar, "Broadcast Encryption," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **773**, pp 480–491, 1994.
[9] D. Noar, M. Noar, J. Lotspiech, "Revocation and Tracing Routines for Stateless Receivers," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **2139**, 2001.
[10] M. Ramkumar, "I-HARPS: An Efficient Key Predistribution Scheme for Mobile Computing Applications," IEEE Globecom, San Francisco, CA, Nov 2006.