Self-Organized Network-Layer Security in Mobile Ad Hoc Networks*

Hao Yang, Xiaoqiao Meng, Songwu Lu
Department of Computer Science
University of California, Los Angeles
{hyang,xqmeng,slu}@cs.ucla.edu

ABSTRACT

Protecting the network layer in a mobile ad hoc network is an important research topic in wireless security. This paper describes our unified network-layer security solution in ad hoc networks, which protects both routing and packet forwarding functionalities in the context of the AODV protocol. To address the unique characteristics of ad hoc networks, we take a self-organized approach by exploiting full localized design, without assuming any *a priori* trust or secret association between nodes. In our design, each node has a token in order to participate in the network operations, and its local neighbors collaboratively monitor it to detect any misbehavior in routing or packet forwarding services. Upon expiration of the token, each node renews its token via its multiple neighbors. The period of the validity of a node's token is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its token less and less frequently as time evolves. In essence, our security solution exploits collaboration among local nodes to protect the network layer without completely trusting any individual node.

Categories and Subject Descriptors

C.2.0 [**Security and Protection**]: Wireless Security

General Terms

Security, Design

Keywords

Self-Organized Security, Mobile Ad Hoc Networks

1. INTRODUCTION

Protecting the network layer in a mobile ad hoc network is an important research topic in wireless security. Without appropriate security solutions, an ad hoc network is exposed to various network-layer security threats, as enumerated by several recent studies [2, 5, 7]. The problem is challenging due to the lack of centralized management/monitoring component, error-prone multi-hop wireless communication, and dynamics in the network topology.

The core functionalities provided in the network layer are routing and packet forwarding. Malicious attacks on either of them will disrupt the normal network operations. Although several recent proposals [18, 6, 5, 11] have addressed the problem of secure ad hoc routing, protection of packet forwarding service has received relatively less attention except the work of [10]. This paper is interested in devising a coherent, unified solution that protects both the routing and the data forwarding services in mobile ad hoc networks.

Most existing security schemes proposed for mobile ad hoc networks either assume *a priori* trust or secret association between networking entities [6, 5, 11, 7], or assume that there is a centralized trusted server in the network [2]. However, the self-organized nature of the ad hoc networks challenges this very basic assumption, and the existence of a centralized server may degrade the effectiveness of the security scheme [9].

This paper describes our work-in-progress solution to the network-layer security in ad hoc networks in the context of AODV [13] routing protocol. Our solution takes a self-organized approach, without assuming any *a priori* trust or secret association between nodes, or the existence of any centralized trusted entity in the network. The self-organized feature of the solution is provided through fully localized design: each node shares a portion of a global secret, and each node is verified and monitored by its local neighbors collaboratively. Fundamentally, our security solution exploits the collaboration among local nodes to protect the network layer without completely trusting any individual node.

In our design, each node is granted temporary admission into the network initially by obtaining a token that will expire soon. Once the token expires, the node has to renew it from its local neighbors, which are responsible for monitoring its behavior collaboratively. The node accumulates its credit as it stays and behaves well in the network. The period of validity of a node's token is proportional to its current credit. This way, a well-behaving node renews its token less and less frequently as time evolves. A malicious node will eventually be detected by its neighbors, its token will be revoked, and it will be denied network access. We also extend the AODV protocol to devise a coherent solution to protect both routing and packet forwarding functionalities.

*This work is supported in part by an NSF CAREER grant (ANI-0093484).

WiSe'02, September 28, 2002, Atlanta, Georgia, USA.

The rest of the paper is organized as follows. Section 2 provides background knowledge on AODV routing protocol and network layer security issues. Section 3 lists our design goals. Section 4 describes our self-organized security solution in details. Section 5 explains our design rationale, and discusses some important issues for future improvement. Section 6 compares our security solution with the related work, and Section 7 summarizes the paper and outlines our immediate future work.

2. BACKGROUND

2.1 AODV Routing Protocol

We consider routing and data forwarding security issues in the context of the Ad hoc On-demand Distance Vector routing protocol (AODV). AODV has been one of the most popular on-demand routing protocols studied in the research community and IETF [1, 3, 14]. For simplicity, we focus on the basic version of AODV of [13] in this work.

In AODV, path discovery is entirely on-demand. When a source node needs to send packets to a destination to which it has no available route, it broadcasts a RREQ (Route Request) packet to its neighbors. Each node maintains a monotonically increasing sequence number to ensure loop-free routing and supersede stale route cache. The source node includes the known sequence number of the destination in the RREQ packet. The intermediate node receiving a RREQ packet checks its route table entries. If it possesses a route toward the destination with greater sequence number than that in the RREQ packet, it unicasts a RREP (Route Reply) packet back to its neighbor from which it received the RREQ packet. Otherwise, it sets up the reverse path and then rebroadcasts the RREQ packet. Duplicate RREQ packets received by one node are silently dropped. This way, the RREQ packet is flooded in a controlled manner in the network, and it will eventually arrive at the destination itself or a node that can supply a fresh route to the destination, which will generate the RREP packet. As the RREP packet is propagated along the reverse path to the source, the intermediate nodes update their routing tables using distributed Bellman-Ford algorithm with additional constraint on the sequence number, and set up the forward path.

AODV also includes the path maintenance mechanism to handle the dynamics in the network topology. Link failures can be detected by either periodic beacons or link layer acknowledgments, such as those provided by 802.11 MAC protocol [4]. Once a link is broken, an unsolicited RREP packet with a fresh sequence number and infinite hop count is propagated to all active source nodes that are currently using this link. When the source node receives the notification of a broken link, it may restart the path discovery process if it still needs a route to the destination.

2.2 Network-Layer Vulnerabilities

Ad hoc networks are vulnerable to a wide range of malicious attacks in the network layer due to the inherent peer-to-peer communication model. In these networks, each node functions as a router that maintains routes toward other nodes in the network, and each node relies on intermediate nodes to relay its packets to the destination. Malicious attacker may readily become a router and disrupt normal network operations.

The core functionalities of the network layer are routing and packet forwarding. Routes from the source to the destination are established and maintained by the routing protocols, while data packets are forwarded by intermediate nodes along the established route to the destination. Attacks on either functionality can disrupt the normal operations in the network layer.

Although routing and packet forwarding functionalities are closely related to each other, we explicitly distinguish their vulnerabilities because the routing functionality is only responsible for establishing and maintaining the routes, and it can not enforce that the data packets are correctly forwarded along the routes by any means. Therefore, we describe the network-layer vulnerabilities by two categories of attacks: *routing updates misbehavior* and *packet forwarding misbehavior*.

Routing updates misbehavior means any action of advertising routing updates that does not follow the specifications of the routing protocol. Because ad hoc routing protocols typically assume that all nodes are cooperative, the attacker may exploit this vulnerability and inject malicious routing information into the network. In the context of AODV, the attacker may advertise a route with a smaller distance metric than its actual distance to the destination; the attacker may advertise routing updates with a large sequence number and invalidate all the routing updates from other nodes; the attacker may also spoof its IP address and advertise that an operational link is broken.

By exploiting routing updates misbehavior, the attacker can attract data traffic to itself, or cause the packets to be forwarded along a route that is not optimal, with poor quality, or even nonexistent. The attackers can also intentionally introduce severe network congestion and channel contention in certain areas. If there are multiple attackers in the network, they may even collaborate to prevent a source node to find any route to the destination, partition the network, or create route loops and waste the network resource.

Packet forwarding misbehavior means any malfunction of the data packet forwarding service as the consequence of an attack. For example, the attacker along an established route may drop the data packets, or duplicate the data packets that it has forwarded. Another type of packet forwarding misbehavior is the Denial of Service (DoS) attack of network layer packet jamming, in which the attacker injects large amount of packets into the network and wastes a significant portion of the network resource. Furthermore, the attacker may adopt more tricky strategies, such as dropping certain data packets or dropping the data packets with some probability, instead of blindly dropping all the packets.

Attacks may be initiated toward each of these two dimensions of routing and packet forwarding, or both. Even though the attacker exactly follows the routing protocol, it can still generate various packet forwarding misbehaviors, such as the network-layer DoS attack.

3. DESIGN GOALS

Our fundamental goal is to provide a *coherent, unified network-layer security solution* to protect both the routing and packet forwarding functionalities in ad hoc networks. Although it is possible to protect each functionality independently [6, 11, 10], an unified solution is desirable. It can avoid the difficulty or complexity of combining different security schemes, and it can benefit from the interactions between routing and packet forwarding. For the solution, we also have four more goals as elaborated below.

First, the security solution should be *self-organized*. The centralized security paradigm, which is popular in wired networks, relies on a centralized trusted entity, such as a TTP (Trusted Third Party) or a KDC (Key Distribution Center), to establish trust relationship between different nodes. However, it does not work well in ad hoc networks due to their unique characteristics. Note that the centralized trusted entity may have to stay online all the time in order to nullify the malicious or compromised nodes. The existence of such a centralized entity may hurt the effectiveness of the security solution because: (1) The centralized entity is prone to DoS attack, which becomes the single point of failure. When DoS is initiated toward the server, the server cannot provide services (e.g., token revocation) to the network. (2) It is nontrivial for a mobile host which is far away from the centralized entity to contact it in a timely manner through multi-hop wireless communications. (3) Since the wireless channel is shared and bandwidth constrained, channel contention and network congestion around the centralized entity will reduce the service availability and increase the access latency.

In our design, we do not assume the existence of any centralized trusted entity in the network, neither do we assume any *a priori* secret association or trust relationship between the nodes. Instead, we take the self-organized approach by adopting fully localized mechanisms and relying on the collaboration in the local neighborhood to detect and prevent malicious attacks.

Second, the security solution should be *tolerant* of the existence of compromised nodes. Unlike the Internet routers which provide only limited services with careful protection, nodes in ad hoc networks are much more vulnerable to compromise. Break-ins due to OS bugs, backdoors, email viruses, may happen occasionally. Mobile hosts without adequate physical protection are also prone to being captured. Therefore, this intrusion tolerance feature is important for the security solution in ad hoc networks. Our design employs the threshold cryptography primitives to enhance its tolerance against the compromised node.

Third, the security solution should *isolate* the attackers and compromised nodes in the network. Compared to avoiding them in selecting routes to forward legitimate traffic, proactively isolating them has more benefit, because it ensures that the attackers cannot continue the attack and waste the network resource in the future. This feature essentially makes our security solution capable to isolate DoS attack in the network layer.

Finally, the security solution should have *decreasing overhead* over time when the network is in good condition without attacks. In general, any security solution will unavoidably introduce extra communication and computation overhead. However, this may become a serious problem in the ad hoc networks consisting of low-end devices. By adopting credit based strategy (Section *4.2.2*), our security solution will cause less and less overhead as the network is in operation, which is well suited for the resource constrained ad hoc networks.

4. NETWORK-LAYER SECURITY SOLUTION

Our design is based on the following assumptions:

- Any two nodes within the wireless communication range may interact with each other over the shared wireless channel. Each wireless interface may operate in the promiscuous mode, i.e., if node A is within the communication range of node B, then node A can overhear all the communications going on at node B.

- Our focus is network-layer security issues. We do not consider physical-layer or link-layer issues, which are also vulnerable to malicious attacks. Attacks against these layers can be limited by lower-layer mechanisms such as the spread-spectrum technology or the WEP protocol.

- Although it is important to protect the data packets from eavesdropping and modification, we do not elaborate on these vulnerabilities due to space limit. Once the source and the destination have established a secure route for data forwarding, they can further exchange a symmetric key and encrypt data packets to ensure data confidentiality and integrity.

- We assume that multiple attackers may coexist in the network. We do not differentiate compromised nodes from attackers from the security point of view. However, in order to make the problem tractable, we limit the power of the attackers: (1) Each node has a unique ID[1]. (2) The underlying cryptography primitives, such as RSA, are practicaly secure; The attackers cannot break them with current computational power; (3) Collaboration among the attackers is limited, i.e., less than k attackers cooperate in any local neighborhood.

4.1 Framework

In order to protect the routing and packet forwarding functionalities in ad hoc networks, our network-layer security solution consists of both proactive and reactive mechanisms. Each legitimate node carries a token signed with the system secret key, which can be verified by its neighbors. Nodes without a valid token are isolated in the network in that all its legitimate neighbors will not interact with them in routing and forwarding services. The system secret is equally shared by all nodes in the network, but each node only knows a limited portion of it. The token has limited period of validity. Before its token expires, each node must renew the token from its neighbors, which in turn collaboratively monitor it to detect any misbehavior. Once an attacker is detected, its token will be revoked, which deprives the attacker of the network access.

Our security solution is fully localized in that all the basic operations are performed in the local neighborhood. Each node monitors the behavior of its neighbors, verifies and issues tokens to its neighbors, and interacts only with its legitimate neighbors. When the attackers are detected in their local neighborhood, all the nodes in the network will be notified through the intrusion reaction mechanism, thus effectively isolating them and preventing them from further

[1]This ID is used to differentiate the nodes in the network, instead of addressing the authentication problem. MAC addresses can serve this purpose because it is hard to change the hardware settings in practice. This is justified by our experience with the Lucent wireless card, which shows that the wireless card performs double-check and enforces to use the embedded MAC address when it sends out a packet.

launching the attack. In essence, our security solution exploits collaboration among local nodes without completely trusting any individual node.

Figure 1 illustrates the composition of our security solution, which consists of four closely interacted components:

- *Neighbor Verification*, which describes how to verify whether each node in the network is a legitimate or malicious node.
- *Security Enhanced Routing Protocol*, which explicitly incorporates the security information into the ad hoc routing protocol.
- *Neighbor Monitoring*, which describes how to monitor the behavior of each node in the network and detect occasional attacks from malicious nodes.
- *Intrusion Reaction*, which describes how to alert the network and isolate the attackers.

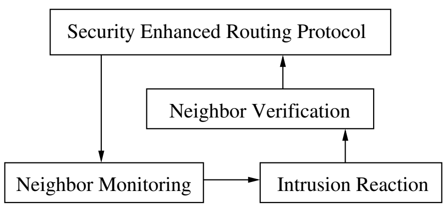

Figure 1: Framework of the network layer security solution

In this framework, neighbor verification and security enhanced routing protocol proactively prevent the attackers from disrupting the network operations; neighbor monitoring detects any misbehavior in both routing and packet forwarding services; and intrusion reaction serves as the bridge between neighbor monitoring and neighbor verification and isolates the detected attackers. In the following sections, we will describe the design of these four components in details.

4.2 Neighbor Verification

The neighbor verification mechanism is based on tokens and employs the asymmetric cryptographic primitives, specifically the *de facto* standard RSA [16]. There is a global secret key pair SK/PK, and PK is known to all nodes when they join the network. Each legitimate node carries a token stamped with an expiration time and signed by SK. The token of a node contains the following three fields <*owner_identity, signing_time, expiration_time*>. Each node periodically broadcasts the token in the *hello* message to its neighbors. Token verification is simple in that a token is valid if and only if 1) it is held by the node with the same identity as stated in the *owner_identity* field; 2) it has not expired; and 3) it is signed by SK. Any node without a valid token will be regarded by its neighbors as a malicious node, and all its packets, both routing updates and data packets, will be dropped.

There are three critical questions regarding how to issue the tokens for the nodes: 1) Who is responsible for issuing the tokens? 2) How do the nodes obtain their tokens? 3) What is the period of validity of each token?

We now consider the first question. In our design, we do not assume any centralized trusted server to issue the tokens for the nodes. Instead, our token issuing process is decentralized, and each node will participate in issuing tokens for its neighbors. However, in order to improve the tolerance against compromised nodes, we rely on the collaboration among the nodes without giving any individual node enough power to disrupt the whole security solution. This is realized by employing the polynomial secret sharing scheme [9, 17], in which each node shares the SK by a polynomial of order $k-1$. Due to limited space, we refer to [9] about the details of this scheme, for example, how the token is collaboratively signed, and how the system is bootstrapped. With this scheme, the token of each node is always issued and signed by its k neighbors. Since a group of k nodes can collaborate to sign a token with SK, but a group of less than k nodes can never sign a token, our design is robust to less than k compromised nodes in the local neighborhood.

After answering the fundamental question of the authority for token issuing, we now address the other two questions and illustrate our localized token issuing process and credit-based expiration timer strategy.

4.2.1 Localized Token Issuing

Consider the case that a node in the network, which already possesses a token, needs to renew its current token. The message handshake in the localized token issuing process is illustrated in Figure 2. Before the expiration time of a node's current token, it broadcasts a TREQ (Token Request) packet to its neighbors, which contains its current token and a timestamp. Each node also keeps a Token Revocation List (TRL) learned from the intrusion reaction component. When a node receives a TREQ packet, the TRL will be used to decide whether to serve the request or not.

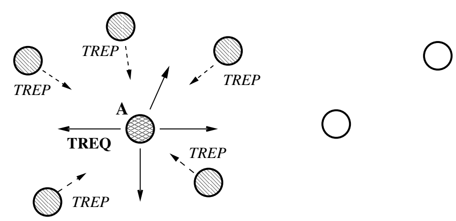

Figure 2: Message handshake in the localized token issuing process

Specifically, when a node receives a TREQ packet from its neighbor, it extracts the token from the packet. It checks whether the TREQ packet comes from the owner of the token therein, and whether the token has already been revoked by comparing it with the TRL. If the token is still valid and the source of the TREQ packet matches the owner of the token, it constructs a new token in which *owner_identity* is equal to that in the old token, *signing_time* is equal to the timestamp in the TREQ packet, and *expiration_time* is determined by the additive increase algorithm described below. It then signs the newly constructed token using its own share of SK, encapsulates the partially signed token in a TREP (Token Reply) packet, and then unicasts the TREP packet back to the node from which it received the TREQ packet. TREQ packets from incorrect sources or containing revoked tokens will be silently dropped. When the node which needs to renew its token receives k TREP packets from different neighbors, it can combine these partially signed tokens into a single token signed by SK.

There is another case of token issuing: a newly joined node needs to obtain its first token. This is similar to the token renewing case from the message handshake perspective. In order to join the network, a node also broadcasts a TREQ (Token Request) packet, containing its identity and the current time, in its local neighborhood. Its neighbors apply the same rules as described above to determine whether to serve this request and, if they decide to issue the token, apply the same process to construct and send back the partially signed tokens. However, the *expiration_time* field in the first token is different from that in a renewed token.

4.2.2 Token Expiration Timer

The expiration timer of a token, i.e., the *expiration_time* field in the token, represents the tradeoff between the computation overhead and the length of TRL. Choosing larger expiration timer will decrease the computation overhead as fewer token renewal process is required; however, it will also increase the expected length of TRL, because once the token is revoked, it will remain in the TRL for longer period of time until it expires.

We adopt credit based strategy in determining the expiration timer of each node's token. The period of validity of a node's current token is dependent on how long it has stayed and behaved well in the network. A newly joined node is issued a token with small period of validity. When it remains to behave well in the network, its subsequent tokens will have longer and longer period of validity. This is achieved by additively increasing the period of validity when a node renews its token from its neighbors.

Let T_1, T_2, T_3, T_4 denote the *signing_time* and *expiration_time* fields in the previous and renewed tokens, respectively. The additive increase algorithm states that

$$T_4 - T_3 = T_2 - T_1 + T_0 \tag{1}$$

By this simple algorithm, each time a legitimate node renews its token, the period of validity of its token increases by T_0. This can significantly reduce the communication and computation overhead when the network is in operation, as the legitimate nodes renew their tokens less and less frequently.

With the assumption that the probability of a node being an attacker is reciprocal to the duration of the time it has stayed and behaved well in the network[2], we can show the benefit of the credit based strategy by comparing it to the constant period of validity strategy, which always sets the period of validity of a token to T_0.

In the credit based strategy, when a node receives its nth token, the duration of the time it has stayed in the network is $T_L = \sum_{i=1}^{n-1} iT_0 = \frac{n(n-1)T_0}{2}$.[3] For a node with lifetime T, by setting T_L equal to T, the total number of token renewal processes can be easily obtained as $N_1 \approx \sqrt{\frac{2T}{T_0}}$. However, in the constant period of validity strategy, the total number of token renewal processes is $N_2 = \frac{T}{T_0}$. We have $N_2 \approx \frac{N_1^2}{2}$, which demonstrates the savings of the credit based strategy in terms of computation and communication overhead.

On the other hand, in the credit based strategy, the expected time of the current token kept in the TRL is

$$\begin{aligned} T_C &= \int_{T_L}^{T_L+nT_0} \frac{T_L + nT_0 - t}{t} \mathrm{d}t \\ &< \int_{T_L}^{T_L+nT_0} \frac{T_L + nT_0 - t}{T_L} \mathrm{d}t = \frac{nT_0}{n-1} \end{aligned} \tag{2}$$

We can see that the expected time of one node's token kept in the TRL is asymptotically bounded by T_0, which shows that the credit based strategy does not impose heavy burden on the length of the TRL.

In order to avoid synchronization among the token renewal requests in the network, we also introduce randomization on the time that a node broadcasts the TREQ packet to renew its token. Let T_s and T_e denote the *signing_time* and *expiration_time* fields in a node's current token, respectively. Instead of requesting token renewal exactly before T_e, the node will randomly pick up a value $\hat{T}_e$ with uniform distribution over $[0.25 * T_s + 0.75 * T_e, T_e]$, and broadcasts the TREQ packet at time $\hat{T}_e$.

4.3 Security Enhanced Routing Protocol

We extend the AODV protocol [13] and explicitly incorporate the security information in our security enhanced ad hoc routing protocol, which we call AODV-S. AODV-S retains most of the AODV mechanisms, such as on-demand path discovery, reverse path setup, forward path setup, soft-state associated with the route entry, path maintenance, local connectivity management. In this section, we will mainly describe the difference between them.

Each AODV-S node maintains the list of all its verified neighbors which possess valid tokens. This can be easily achieved by taking advantage of the local connectivity management in AODV and the neighbor verification mechanism described earlier. Each AODV-S node only interacts with its verified neighbors. All the routing updates received from a neighbor without a valid token will be dropped.

One possible approach to prevent routing updates misbehavior is to encrypt or attach Message Authentication Code (MAC[4]) to all routing updates. However, we do not take this approach due to several considerations.

First, in distance vector routing protocols, the routing information is compressed into several routing metrics, such as hop count and destination sequence number in AODV. Each node disseminates routing updates on its own will, and each routing update is only directly visible to the neighbors of the sender, as opposed to source routing protocols. Neither encryption nor MAC based on one node's own secret key can prevent compromised nodes to disseminate malicious routing updates. Second, encryption or MAC based on the source-destination pairwise secret key requires that each pair of nodes share a secret key (in the symmetric cryptography), or each node has the public keys of all the other nodes (in the asymmetric cryptography), which can be hardly achieved in the dynamic ad hoc networks without a centralized key management service. Third, encryption/decryption of the

[2]This assumption is motivated by the analogy of how the credit card companies set up the credit line for their customers. We admit that it is a simplified model for the user behavior. However, it indeed reflects some characteristics of the attackers in that they usually will not stay and behave well in the network for a long time.

[3]For simplicity of representation, we assume that the period of validity of the first token is T_0.

[4]In this paper, we use MAC to represent the Message Authentication Code instead of the link layer Medium Access Control protocols.

routing updates causes significant computation load, and may be utilized by the attackers to launch DoS attack.

Instead, we rely on the redundancy of the routing information to prevent routing updates misbehavior. The basic idea is that each node explicitly claims the next hop node when it disseminates a new routing update, and each node keeps track of the route entries previously announced by its neighbors. In this way, each node can maintain part of the routing tables of its neighbors. This redundancy of the routing information makes it possible for a node to examine the correctness of routing updates, because the execution of the distributed Bellman-Ford algorithm should be based on the route updates previously disseminated by some neighbors, which this node may also have received.

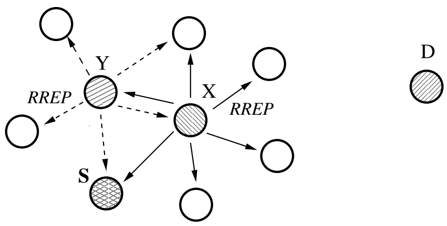

Figure 3: Using redundant routing information to examine the correctness of routing updates

Figure 3 illustrates how the routing updates are examined based on the redundant routing information. Node S is the neighbor of both node X and node Y. S has kept track of the route entries previously announced by Y. When S receives a new routing update from X, and the next hop claimed by X is Y, it can examine the correctness of this routing update by comparing the new route entry with the corresponding route entry previously announced by Y. We can view this process as that S is reconstructing the execution of the distributed Bellman-Ford algorithm performed by X.

Specifically, we add one more field, *next_hop*, in the RREP packet, which means that the RREP packet in AODV-S contains six fields <*source_addr, dest_addr, dest_sequence_#, hop_cnt, next_hop, lifetime*>. We also modify the way of propagating RREP packets. Each AODV-S node *broadcasts* the RREP packets to its neighbors, as opposed to unicasting the RREP packets along the reverse path in AODV. Finally, in addition to its own routing table, each AODV-S node also maintains the route entries announced by its verified neighbors. An announced route entry contains the following fields <*neighbor_addr, dest_addr, dest_sequence_#, hop_cnt, next_hop, lifetime*>.

In AODV-S, when a node receives a RREP packet, it first examines the correctness of the routing update, using the simple algorithm described in Section *4.4.1*. Incorrect RREP packets will be dropped. If the routing update in the RREP packet is correct, it updates its own routing table in a way similar to AODV, and updates its cache of the announced route entries of the corresponding neighbor. A node receiving a RREP packet also checks whether it previously sent the corresponding RREQ packet. If so, it rebroadcasts the RREP using its updated route table entries. In this way, the RREP packets will be propagated back to the source.

4.4 Neighbor Monitoring

In the neighbor monitoring mechanism, each node is responsible for monitoring the behavior of its neighbors and detecting any misbehavior in both routing and packet forwarding services. Those misbehavior will be regarded as indications of attacks. Furthermore, all the nodes in one neighborhood collaborate with each other to improve the accuracy of monitoring results and withstand sophisticated attacks.

4.4.1 Monitoring Routing Updates Misbehavior

The routing updates misbehavior is detected by examining the correctness of routing updates. When a node receives a RREP packet broadcasted by its verified neighbors, it first examines the correctness of the newly offered route. We consider the example scenario in Figure 3 again, in which S and X are the receiver and sender of the RREP packet, respectively, while D and Y are the destination and the next hop specified in the RREP packet, respectively.

If S is also the neighbor of Y, it compares the new route entry offered by X with its cached route entry previously announced by Y and destined to D. The new route entry is correct if and only if the sequence number in the two route entries are the same, and the hop count in the new route entry is one larger than the hop count in the cached route entry announced by Y. If the routing update is not correct, the RREP packet is dropped and node S broadcasts a SID(Single Intrusion Detection) packet to its neighbors. Note that it is also possible for S to be out of the neighborhood of Y. In this case, S will skip this examination process, because S has no information about the next hop node in the offered route.

The routing updates examination algorithm has some weaknesses in that it might not work well in several situations: 1) Y only stayed in S's neighborhood for a short period of time due to mobility, so that S has not recorded all the route entries announced by Y; 2) S did not receive the previous route updates broadcasted by Y due to channel error and contention; 3) Y has increased the lifetime of a route entry, but S is not aware of this change and has deleted it from its cache. It is also susceptible to the *blackmail* attack, in which an attacker blackmails its legitimate neighbors as misbehaving nodes. However, we rely on the collaborative monitoring mechanism (Section *4.4.3*) to improve the monitoring accuracy and withstand the *blackmail* attack.

4.4.2 Monitoring Packet Forwarding Misbehavior

In addition to monitoring routing updates misbehavior, each node also monitors its neighbors to detect misbehavior in data packet forwarding service. This can be done in ad hoc networks through overhearing the channel in promiscuous mode in 802.11 link layer.

We currently consider three kinds of packet forwarding misbehavior, namely, packet dropping, packet duplicating, and network layer packet jamming, and develop simple algorithms for each of them. Packet dropping means that a node drops the packets that it is supposed to forward for its neighbors; packet duplicating means that a node duplicates the packets that it has already forwarded; and network layer packet jamming means that a node sends too many packets and occupies a significant portion of the bandwidth.

The packet dropping detection algorithm is similar to the *watchdog* technique in [10]. The *watchdog* was originally proposed for DSR, in which the sender explicitly list the route in the data packet header. It can not be directly applied in AODV, because if one node receives a packet, its neighbors do not know which node it should forward the packet to, and can not tell whether it has forwarded the packet in the correct manner. However, the *watchdog* can be extended to work with AODV-S, because each AODV-S node keeps track of the route entries announced by its neighbors, which explicitly include the *next_hop* field.

Specifically, each node overhears the channel at all time and records the headers of the recent packets it has overheard. If it overhears one packet sent to its neighbor, say, X, for forwarding, it checks its cache of the route entries announced by X and determines the next hop node to which X should forward the packet. If it does not overhear the packet being forwarded by X to the correct neighbor after *Drop_Time* seconds, it considers this packet to be dropped. If the bandwidth corresponding to the packets dropped by X exceeds the threshold *Drop_Bandwidth*, it considers X as an attacker and broadcasts the SID (Single Intrusion Detection) packet.

The packet duplicating and packet jamming detection algorithms also utilize the information obtained by overhearing the channel. If one node overhears that the bandwidth corresponding to the duplicate forwarding of packets by its neighbor X exceeds the threshold *Duplicate_Bandwidth*, or the bandwidth corresponding to the packets sent by its neighbor X exceeds the threshold *Sending_Bandwidth*, it considers this as the indication of an attack and broadcasts the SID packet.

The localized monitoring mechanism executed by each node is intrinsically inaccurate due to the inaccuracy in the information obtained by overhearing the channel. The detection accuracy is also sensitive to multiple factors, such as channel error, mobility, parameters in the detection algorithm, etc. Next we will describe the distributed collaborative monitoring mechanism in the local neighborhood, which can significantly improve the monitoring performance.

4.4.3 Distributed Collaborative Monitoring

In order to improve the monitoring accuracy and withstand the *blackmail* attack, we use "m out of N" strategy to cross-validate the monitoring results of different nodes in one neighborhood. That is, a node is considered as an attacker if and only if m nodes out of all its N neighbors have independently sent out SID packets against it.

The "m out of N" strategy can significant improve the accuracy of monitoring results, which can be quantitatively evaluated by two metrics: the probabilities of Class I error (failure to detect the attacker) and Class II error (false accusation against a legitimate node). Let P_1 and P_2 denote the probabilities of Class I error and Class II error in the monitoring result made by a single node, respectively. By this collaborative monitoring, the detection probability for an attacker is:

$$P_D = \sum_{k=m}^{N} \binom{N}{k} (1-P_1)^k \; P_1^{N-k} \tag{3}$$

Meanwhile, the false detection probability for a legitimate node is:

$$P_F = \sum_{k=m}^{N} \binom{N}{k} P_2^k \; (1-P_2)^{N-k} \tag{4}$$

From Figure 4 and 5 we can see that by choosing appropriate value for m, we can increase P_D and decrease P_F simultaneously. There are several approaches to determine m as a function of N, such as setting m as $N/2$, setting m as k (the secret sharing parameter), or setting m to guarantee that both P_D and P_F are within certain range. The selection of m represents the tradeoff between the prompt reaction to the attackers and the protection of legitimate nodes from false accusation. We are currently exploring the impact of different schemes, which is out of the scope in this paper.

We take advantage of the polynomial secret sharing scheme again to implement the collaborative monitoring mechanism. Note that we do not differentiate the SID packets triggered by the routing updates misbehavior and the packet forwarding misbehavior. When a node has received m independent SID packets against the same node, it constructs a notification of token revocation, signs the notification using its own share of SK, encapsulates the signed notification in a GID(Group Intrusion Detection) packet, and then broadcasts the GID packet. The first node that receives k GID packets against the same node combines them and constructs a TREV (Token Revocation) packet which is signed by the SK, based on the polynomial secret sharing cryptography primitives. In the next section, we will describe the mechanism to propagate the TREV packet and isolate the attackers in the network.

4.5 Intrusion Reaction

The intrusion reaction mechanism serves as the bridge between neighbor verification and neighbor monitoring. Recall that each node keeps a TRL (Token Revocation List). When a node receives a TREV packet, it checks whether the packet is signed by SK, and whether the revoked token is already on the TRL. TREV packet that is not signed by SK or contains a token on the TRL is silently dropped. Otherwise, it adds the token into the TRL and then rebroadcasts the TREV packet. In this way, eventually every node will add the revoked token into its TRL. Meanwhile, the neighbors of an attacker deem the links between them and the attacker that are currently in use as broken, and use the path maintenance mechanism in the routing protocol to cancel out these links.

Each entry in the TRL is associated with a *lifetime*, which is equal to the *expiration_time* in the corresponding token. When the token expires, none of the nodes needs to maintain this revocation information. The soft-state associated with TRL entries can reduce both the storage overhead and the checking overhead when a node receives the token renewal requests from its neighbors.

Recall that each node only interacts with verified neighbors. The intrusion reaction mechanism guarantees that the attacker is isolated in the network right after it is detected, and it will never be issued a new token again in the future. Although the TREV packet is flooded in the network, the communication overhead is still affordable, because the intrusion reaction process is triggered only once for each attacker or compromised node.

5. DISCUSSION

5.1 Design Rationale

We now come back to elaborate on several design choices we have made in our security solution, namely, asymmetric

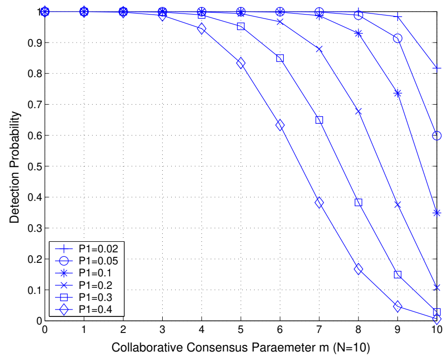

Figure 4: Improvement on increasing the detection probability by collaborative monitoring

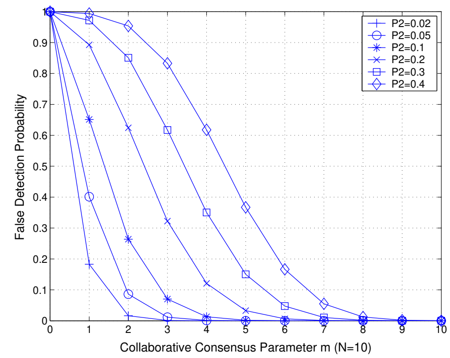

Figure 5: Improvement on decreasing the false detection probability by collaborative monitoring

cryptography primitive, localized token issuing, localized intrusion detection, and global reaction to the attackers.

Cryptography Primitive The asymmetric cryptography primitive (RSA) we use in the security solution has relatively high computation overhead, compared to the symmetric cryptography primitives. We justify this design choice by four reasons. First, we pursue a self-organized security scheme that does not assume any *a priori* secret association between the nodes or the existence of any centralized trusted entity. In this case, the key management, which is the fundamental component in any security scheme exploiting symmetric cryptography primitives, is extremely difficult. Second, by adopting the credit based strategy for the token expiration timer, the overhead of our security solution will decrease over time when the network is in operation. We deem this a nice feature for a self-organized security scheme. Third, in our security solution, only tokens and the TREV (Token Revocation) packets are encrypted, and the encryption only happens when one token is issued or an attacker is detected. We indeed avoid the overhead of encrypting all the routing updates. Lastly, we believe that the mobile devices, even the low-end devices, will have more and more computation power with the continuous development of hardwares.

Token Issuing In our localized token issuing mechanism, each node shares the system key SK by a polynomial of order $k-1$, and k neighbors collaborate to sign one token. We make this design choice due to several considerations. First, since we do not assume any centralized trusted entity, each node has to obtain the signed tokens from some peer nodes. We choose its neighbors to issue the token to avoid multi-hop communication over error-prone wireless channel, which may decrease the service availability and increase the service delay. Second, secret sharing can significantly improve the tolerance against the compromised nodes. Our security solution works well with up to $k-1$ compromised nodes in one local neighborhood.

Intrusion Detection In our localized intrusion detection mechanism, each node monitors its neighbors to detect any misbehavior. This is because there is no traffic concentration point in ad hoc networks. An alternative approach is the end-end-end intrusion detection scheme, in which the sender detects the quality of the route based on the feedback from the receiver. However, this approach can only determine whether there is any attacker in the route, instead of which node is an attacker. It also incurs extra communication overhead when the transport layer protocols, such as UDP, and the application layer protocols do not provide any feedback from the receiver to the sender.

Intrusion Reaction Our intrusion reaction mechanism guarantees that the attacker is isolated in the network once it is detected by its neighbors. This can be viewed as a global reaction scheme. An alternative approach is the end-to-end reaction scheme, in which the sender tries its best to avoid the attackers that it is aware of. This scheme is often combined with the end-to-end intrusion detection scheme. However, we abandon this scheme due to several reasons. First, while it works well in the source routing protocols, it is difficult to be extended to work with distance vector routing protocols, because once the sender pumps the packets into the network, it can not control the route along which the packets will be forwarded. Second, it is desirable to proactively isolate the attackers, so that they can not continue the attack and waste the network resource in the future.

5.2 Related Issues

In this section, we will enumerate a few unsolved problems in our current design and discuss the future improvements to address these problems.

Collaboration Among Attackers We assume that the collaboration among the attackers is limited, i.e., only attackers in the same neighborhood can collaborate with each other. We comment that more powerful collaboration among the attackers will decrease the security strength of our solution. We will exploit several strategies, for example, re-keying of the SK, multiple SKs for different neighborhoods, to accommodate more general attack model in the future work.

Energy Efficiency Our localized monitoring mechanism requires that each node overhears the channel all the time. This will cause significant energy consumption. One possible extension is to make each node periodically wake up and undertake the monitoring responsibility, which trades-off between full strength monitoring and energy efficiency. We plan to pursue this direction in the future.

Solution Complexity The overall complexity is another concern about our security solution. We consider this problem from two perspectives: computational complexity and storage overhead. The computational complexity is mainly introduced by the asymmetric cryptography primitives, and the majority of storage overhead is caused by the monitoring mechanism. While modern labtops are able to meet these requirements, it might not be true for low-end devices, such as PDAs. Our future research includes employing alternative light-weighted cryptography primitives to decrease the computation complexity and exploiting hashing techniques to decrease the storage overhead.

Node Density Our security solution relies on the collaboration of the local neighboring nodes in both token issuing and monitoring mechanisms. Sparse node density and high mobility will have negative impact on our design. This problem can be alleviated by seeking collaborations in 2-hop neighborhood in these extreme cases. However, we admit that a more careful study is needed and we leave it for the future research.

6. RELATED WORK

Several researchers have recently studied the problem of secure ad hoc routing [20, 2, 18, 6, 5, 11]. Zhou and Haas [20] proposed a secure routing protocol which exploited threshold cryptography and relied on n secret-sharing servers to protect the routing information. Dahill and others [2] presented the ARAN protocol which used public-key cryptographic certificates for the end-to-end authentication and shortest path confirmation. Our localized verification and token issuing mechanisms differ from these proposals in that we do not assume any centralized trusted server. Yi and others [18] quantified the notion of trust and explicitly incorporated the integrated security metric in the routing protocol. However, their assumption of the *a priori* trust hierarchy obtained by mirroring the organizational hierarchy might not hold in a generic ad hoc network.

Hu and others [6] proposed the Ariadne protocol, which used TESLA [15] one-way key chains and source-destination pairwise keys to protect the DSR [8] routing protocol. The same authors [5] also proposed the SEAD protocol which used the one-way hash chains to secure the DSDV [12] routing protocol. Papadimitratos and Haas [11] presented SRP protocol which relied on the secret association between the source and the destination to protect the source routing messages. All of them assumed some kind of *a priori* secret association or key exchange between the nodes, while our self-organized security solution does not make such an assumption. Another difference between our work and these secure ad hoc routing protocols is that we seek to protect both the routing and the packet forwarding functionalities in the network layer.

There have been some papers on the self-organized security in the ad hoc networks [7, 9]. Hubaux and others [7] proposed the self-organized public-key infrastructure for the ad hoc networks, which was similar to the PGP [21]. In this public-key infrastructure, the certificate of each node is issued by other nodes, and the certificate chain is used to verify a certificate. However, there is an implicit assumption that each node does not make mistakes in issuing the certificates, which made their security design intolerant of the compromised nodes. Kong and others [9] presented the localized authentication scheme for the ad hoc networks. Our work differs from theirs in that we have both the reactive and the proactive components.

Zhang and Lee [19] were among the first to study the problem of intrusion detection in wireless ad hoc networks. Marti and others [10] proposed two techniques, *watchdog* and *pathrater*, to deal with the non-cooperative nodes in the ad hoc networks. Its basic idea of local measurement-based decision is similar to the monitoring mechanism in our security solution. However, the fundamental difference between our intrusion detection mechanism and the *watchdog* is that we utilize the collaborative monitoring strategy to handle the imperfectness in the information obtained by overhearing the channel and significantly improve the detection accuracy.

7. CONCLUSION AND FUTURE WORK

One fundamental challenge for the security design in mobile ad hoc networks is that such networks do not possess any pre-existing infrastructure support. Therefore, the security solution should be provided in a distributed manner. This work explores the self-organized security design for the ad hoc networks. To this end, we have presented an unified network-layer security solution that protects both routing and packet forwarding functionalities. Some nice features of our solution include fully localized design, easy support of dynamic node membership, limited intrusion tolerance capacity (i.e., tolerant of up to $k-1$ collaborative attackers), decreasing overhead over time. While these properties are appealing, we would like to point out that this is achieved at the increased computational overhead (associated with asymmetric cryptography primitives) compared with other hash function based designs [5, 6].

Our ongoing work includes a detailed simulation evaluation of the proposed security solution in terms of message overhead, security analysis, and tolerance to mobile attackers. Another important direction is to compare this design with several recent protocols such as [5, 6, 11]. We expect such results to be available soon in our future report.

8. ACKNOWLEDGMENT

We thank Haiyun Luo and Petros Zerfos for their useful comments on earlier drafts of this paper. We also thank the anonymous reviewers for their valuable comments and suggestions.

9. REFERENCES

[1] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM MobiCom*, 1998.

[2] B. Dahill, B. Levine, E. Royer, and C. Shields. A secure routing protocol for ad hoc networks. Technical Report UM-CS-2001-037, CS Dept., UMass, 2001.

[3] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. IEEE INFOCOM*, 2000.

[4] *IEEE* Standard. Wireless LAN media access control (MAC) and physical layer (PHY) specifications. 1999.

[5] Y. Hu, D. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proc. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2002.

[6] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. Technical Report TR01-383, CS Dept., Rice, 2001.

[7] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proc. ACM MobiHOC*, 2001.

[8] D. Johnson, D. Maltz, and J. Jetcheva. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Network*, chapter 5. Ad Hoc Networking. Addison-Wesley, 2001.

[9] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing robust and ubiquitous security support for manet. In *Proc. IEEE ICNP*, 2001.

[10] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proc. ACM MOBICOM*, 2000.

[11] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proc. SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, 2002.

[12] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proc. ACM SIGCOMM*, 1994.

[13] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[14] C. Perkins, E. Royer, and S. Das. Ad hoc on demand distance vector (aodv) routing. Internet Draft, draft-ietf-manet-aodv-10.txt (Work in Progress), 2002.

[15] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secure source authentication for multicast. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2001.

[16] R. Rivest, A. Shamir, and L. Adeleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[18] S. Yi, P. Naldurg, and R. Keavets. Security-aware ad-hoc routing for wireless networks. Technical Report UIUCDCS-R-2001-2241, UIUC/CS, 2001.

[19] Y. Zhang and W. Lee. Intrusion detection in wireless ad hoc networks. In *Proc. ACM MOBICOM*, 2000.

[20] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Networks*, 13(6):24–30, 1999.

[21] P. Zimmermann. *The official PGP User's Guide*. MIT Press, 1995.