SEAR: A Secure Efficient Ad Hoc On Demand Routing Protocol for Wireless Networks

Qing Li
Rutgers University
qingli@winlab.rutgers.edu

Meiyuan Zhao
Intel Corporation
meiyuan.zhao@intel.com

Jesse Walker
Intel Corporation
jesse.walker@intel.com

Yih-Chun Hu
UIUC
yihchun@crhc.uiuc.edu

Adrian Perrig
CMU
adrian@ece.cmu.edu

Wade Trappe
Rutgers University
trappe@winlab.rutgers.edu

ABSTRACT

Multi-hop routing is essential to the operation of wireless ad hoc networks. Unfortunately, it is very easy for an adversary to forge or modify routing messages to inflict severe damage on the underlying routing protocol. In this paper, we present SEAR, a Secure Efficient Ad hoc Routing protocol for ad hoc networks that is mainly based on efficient symmetric cryptography, with asymmetric cryptography used only for the distribution of initial key commitments. We show, through both theoretical examination and simulations, that SEAR provides better security with significantly less overhead than other existing secure AODV protocols.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Security and protection

General Terms

Security, Performance

Keywords

Authenticator, one-way hash function, SEAR

1. INTRODUCTION

The Ad hoc On-demand Distance Vector routing protocol(AODV) [3] is one of the more popular routing algorithms for MANETs and mesh networks. Unfortunately, providing a secure and trustworthy version of AODV has been elusive. Current secure AODV protocols, such as SAODV [7] and ARAN [6], have significant security weaknesses. For instance, both SAODV and ARAN provide message authenticity only when all intermediate nodes are trustworthy, which is an overly restrictive assumption.

The nature of the AODV protocol has made it challenging to verify routing messages without knowledge of other nodes on the path. In AODV, the optimum routes are selected by choosing the path with the largest sequence number and smallest hop counts. Therefore, the adversaries can easily achieve the routing disruption goal by manipulating the sequence number and hop count fields in the routing messages in such a way that these false routing messages appear to be a better path to reach the destination.

In this paper, we present a secure AODV protocol, which we call the SEAR (Secure Efficient Ad hoc Routing) protocol, that uses one-way hash functions to construct authenticators associated with each node. Route error messages are protected through a variation of the broadcast authentication scheme TESLA [5]. SEAR is the first secure AODV protocol to provide comprehensive solutions to securing both sequence numbering and hop counts simultaneously. Further, SEAR mainly involves highly efficient symmetric cryptography and requires asymmetric cryptography only in the initial bootstrap phase. Compared to existing secure AODV routing protocols, SEAR provides better security with significantly less overhead.

ASIACCS '08, March 18-20, Tokyo, Japan

2. RELATED WORK

Secure routing in ad hoc networks has become an increasingly important topic, and many routing protocols have been proposed to secure ad hoc networks under different attack models [1, 2]. A brief summary of several notable works follows. Hu et. al proposed the Secure Efficient Ad hoc Distance vector routing protocol (SEAD) [1], which is based on the Destination Sequenced Distance Vector (DSDV) [4] routing protocol. An on-demand secure routing protocol, known as Ariadne [2], was also proposed to secure DSR.

There are two secure routing protocols proposed to address the vulnerabilities in AODV algorithms, SAODV [7] and ARAN [6]. Zapata et. al. proposed Secure AODV (SAODV) [7] to protect routing messages transmitted in AODV. Two mechanisms were incorporated into AODV to secure routing messages: digital signatures to authenticate non-mutable fields and hash chains to secure mutable fields. Sanzgiri et. al. designed another secure AODV algorithm, Authenticated Routing for Ad hoc Networks (ARAN) [6]. Similar to SAODV, each node has a certificate signed by a trusted certificate authority. ARAN achieves security via the usage of signatures on a hop-by-hop basis.

3. THREAT ANALYSIS FOR AODV

AODV does not define any security mechanisms, which makes a network based protocol vulnerable to numerous threats. These threats include AODV message forgery and modification attacks. Other vulnerabilities that are not specific to AODV and can be combated through generic techniques. In this section, we will discuss different threats that we face while securing AODV.

Outright forgeries are the most obvious threat against AODV. Since AODV defines no message authenticity mech-

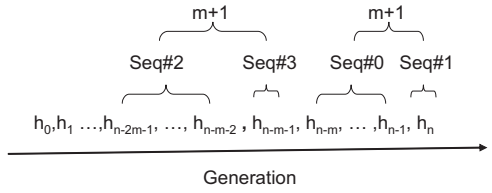

Figure 1: SEAR authenticator hash chain.

anisms, an adversary can forge AODV RREQs, RREPs or RERRs on behalf of other nodes. A related problem is that an intermediate node can alter a legitimate AODV message. For instance, attackers can affect the routing by decreasing the AODV hop count or increasing sequence numbers.

The proposed SEAR protocol addresses all above attacks against wireless ad hoc networks. The traditional philosophy to protect the modification of the routing messages is to apply authentication to prevent the alternation of all fields included in the routing messages. In contrast, SEAR allows for the alteration of certain fields in routing messages, but it guarantees that such an alteration will not result in any benefit to an adversary.

4. SEAR PROTOCOL

SEAR provides routing message authenticity via the use of one-way hash functions to construct a set of hash values, called authenticators, associated with each node. RERR messages are protected through a variation of TESLA [5].

4.1 Assumptions

We assume that each node A has a pair of public/private keys K_A, K_A^{-1}. In order to protect RERR messages, we assume even sequence numbers are used for messages that originate from the destination, while odd sequence numbers are used for those messages that originate from other nodes [4]. We will employ a variation of TESLA to provide authenticity verification of RERR messages. Hence, we assume that all nodes are loosely time synchronized with their neighbors.

We also assume that attackers do not control a set of nodes that partition the network. Finally, for the purpose of providing route freshness and authenticity, we assume that caching mechanisms are disabled on all nodes.

4.2 Initial Bootstrapping

We require that each node maintains two hash chains in SEAR. Firstly, each node constructs an authenticator hash chain to protect the sequence numbers and hop counts for routing packets associated with routes to that node. Secondly, each node creates a TESLA key chain for authenticating RERR messages. These two chains need to be created, and their initial key value commitments need to be delivered securely during the bootstrap procedure.

We begin by providing a high-level overview of the usage of the authenticator hash chain. For discussion, let us denote m to be the maximum hop count that exists between any two network nodes, and $n+1$ to be the length of the hash chain. The authentication hash chain $\{h_0, h_1, \cdots, h_n\}$ is generated using a one-way hash function, H, via the recursion $h_{j+1} = H(h_j)$, with h_n corresponding to the initial commitment for this hash chain. The values of the hash chain, which are called authenticators, are partitioned and used according to the sequence numbers and hop counts associated with the routing messages they protect. A consequence of the one-way function and the partitioning of the authenticators is the fact that other nodes with smaller even sequence numbers will not be able to generate hashes for higher even sequence numbers. For authenticators associated with each even sequence number, other nodes with higher hop counts will not be able to generate hashes for lower hop counts. In SEAR, each individual hop for each even sequence number has a corresponding hash value. In contrast, for RERR messages, we use odd sequence numbers and, since they do not contain a hop count field, we only associate one hash value for each odd sequence number. In order to allow nodes to transmit RERR messages, any node other than the destination that has an even sequence number should also have the corresponding next higher odd sequence number so that it may transmit a RERR message. Consequently, authenticators are broken down into groups of $m+1$ consecutive hash values. These groups of $m+1$ hash values will be associated with an even and odd sequence number in a manner that is the reverse of the hash chain generation procedure. For a specific group of $m+1$ values, one hash value will be assigned to each hop associated with an even sequence number. As noted, the remaining one hash value is associated with the odd sequence number and is used in authenticating RERR messages. In order to follow the above discussion, we present a diagram that depicts the typical structure of the hash chain in Fig. 1.

Attackers cannot decrease the hop count field, but they can still attempt to commit another type of fraud where they transmit or forward routing messages they receive directly, without incrementing the hop count field. In order to prevent such "same hop count fraud", the node identity should be encoded into the hash values to form an authenticator hash tree, as was used in SEAD [1]. Consequently, each node cannot forward routing messages with authenticators encoded with another node's identity, and they must increase the hop count of RREQs and RREPs. We note that for small networks, each node can encode its identity into the hash tree directly, and no adversary can derive its value from neighbors' hash values that correspond to the same hop count. For larger networks, we use a technique similar to in SEAD [1], where γ-tuple values are used to encode node identities into hash values. Now instead of having a unique leaf hash value for each node, each node will have γ leaf hashes.

In order to simplify the description of SEAR in the remaining part of this section, we will only include a discussion involving the one-way hash chain authenticators.

4.3 Securing Route Discovery

Authenticators can be incorporated into RREQ messages to protect the originator sequence number, destination sequence number and hop count. In AODV, whenever the source needs a route, it broadcasts a RREQ:

$$S \rightarrow All, \{S, D, ID, SrcNum, DstNum, hop\}$$

where S is the source address, D is the destination address, ID is the RREQ id, $SrcNum$ is the source sequence number, $DstNum$ is the last destination sequence number known to the source, and hop is the number of hops to the source traversed so far.

Let $v_{s,i,c}$ denote the hash authenticator for node S, with sequence number i and hop count c. Assume that the next even sequence number for the source is $2i$, and the most recent destination sequence number known by the source is j with hop count c. The source thus sends out

$$S \rightarrow All, \{S, D, ID, 2i, v_{s,2i,0}, j, v_{d,j,c}, 0, HERR\}$$

where $HERR$ is used in authenticating RERRs, which will be further discussed in Section 4.5. Each neighbor A will first check the authenticity of the authenticators $v_{s,2i,0}$ and

$v_{d,j,c}$ with the authenticator commitments. If the verification fails, the routing message is ignored. Otherwise, A applies the hash function on the authenticator $v_{s,2i,0}$ to obtain the corresponding one hop authenticator $v_{s,2i,1}$. If A has a larger destination sequence number j' with hop count c', it will replace $v_{d,j,c}$ with $v_{d,j',c'}$. Additionally, A replaces $HERR$ with its own $HERR'$. A thus forwards,

$$A \rightarrow All, \{S, D, ID, 2i, v_{s,2i,1}, j', v_{d,j',c'}, 1, HERR'\} \quad A\ has\ a\ larger\ DstNum$$
$$A \rightarrow All, \{S, D, ID, 2i, v_{s,2i,1}, j, v_{d,j,c}, 1, HERR'\} \quad Otherwise.$$

The procedure repeats until the RREQ reaches the destination or the TTL expires.

Authenticators for even, higher sequence numbers can only be released by the node with hop count 0, thus other nodes cannot release a new, even sequence number or same even sequence number with smaller hop count.

4.4 Securing Route Establishment

The sequence number and hop count in RREP messages can be secured in a manner similar to the approach taken to secure these fields in RREQ messages. Lifetime is usually a system-wide configuration parameter, and hence we don't need to explicitly secure it. The destination or intermediate nodes that have valid routes with equal or higher sequence numbers unicast

$$D \rightarrow S, \{S, D, DstNum, hop, lifetime\}$$

to the originator of the RREQ. For security purposes and to provide freshness of routes, we assume that caching mechanisms are disabled on all nodes. Therefore, only the destination can initiate RREPs in SEAR. Assume the next even sequence number of the destination is $2j$, then the destination unicasts the following RREP to the originator,

$$D \rightarrow S, \{S, D, 2j, v_{d,2j,0}, 0, lifetime, HERR\},$$

where $HERR$ is used to authenticate RERR messages, which will be discussed in section 4.5. Neighbor A will first check the authenticity of the authenticator, then it will apply the one-way hash function to the authenticator $v_{d,2j,0}$ to get $v_{d,2j,1}$. It then replaces $v_{d,2j,0}$ with $v_{d,2j,1}$. Additionally, A replaces $HERR$ with its own $HERR'$. A forwards,

$$A \rightarrow S, \{S, D, j, v_{d,2j,1}, 1, lifetime, HERR'\}.$$

The procedure repeats until RREP reaches the originator.

4.5 Securing Route Maintenance

Once a node detects a link failure for the next hop or receives a RERR from a neighbor for one or more active routes, a RERR message is generated and broadcasted to all upstream neighbors.

The original format of a RERR created by node A is

$$A \rightarrow upstream\ neighbors, \{D - list, DstNum - list\}$$

with $D - list$ describing the list of unreachable destination addresses and $DstNum - list$ describing the corresponding destination sequence number list with each sequence number one larger than the newest even destination sequence number known by A.

First let us look at how the $HERR$ fields attached in the RREQ and RREP messages are created. Assume that the last even source sequence number before forwards RREQs for originator node D, or last even destination sequence number before forwards RREPs for destination node D known by A is $2j$, A can derive the next odd sequence number authenticator $v_{d,2j+1,0}$ via applying the one-way hash function. A will form the RERR message for this sequence number,

$$RERR' = \{Nonce, D, 2j + 1, v_{d,2j+1,0}\}.$$

Let $k_{A,t}$ denote the TESLA key for node A used in time interval t. Then A attaches

$$HERR = \{H(RERR'), MAC_{k_{A,t}}(H(RERR'))\}$$

to the RREQ or RREP messages and saves the nonce and TESLA key in the error messages for later use. When nodes receive the RREQs and RREPs, they buffer the $HERR$ contained in the messages. Further, they replace the $HERR$ in the messages with their own $HERR'$ and save the corresponding information while forwarding RREQ and RREP messages.

Later, when A detects a link failure, or receives a RERR from a neighbor for one or more active routes, it will send a RERR for all unreachable destinations sharing the same next hop with the saved nonce list, corresponding TESLA release key list and destination sequence number list in the same RERR message. If any of the TESLA keys are not ready to be released yet, A will hold the RERR until it can release all the keys. The new format of RERR is,

$$\begin{aligned} RERR \quad = \quad & \{Nonce - list, TESLAkey - list, \\ & D - list, (2j + 1) - list, v_{d,2j+1,0} - list\}. \end{aligned}$$

The upstream nodes extract the information needed from the various lists to reconstruct the $HERR$ and to verify the RERR instantaneously.

5. ANALYSIS AND DISCUSSIONS

In this section, we first provide a comparison of SEAR's performance and security versus other existing secure AODV routing protocols. Then we present a concrete comparison through $ns2$ simulations.

5.1 Performance and Security Evaluation

We will compare SEAR with ARAN and SAODV in terms of their protocol efficiency and security characteristics.

Computation Efficiency: Both of ARAN and SAODV require the originator to sign each packet it sends, and intermediate nodes to verify the signature for each routing packet it processes. On the other hand, SEAR is mainly based on symmetric cryptography. It only requires the originator and intermediate nodes to apply a computationally efficient one-way hash function to generate and verify authenticators.

Communication Overhead: In ARAN, variable number of certificates and signatures are added to all routing messages. In SAODV, there are several hashes and various number of signatures introduced in RREQ/RREP single and double signature extensions and RERR. In contrast, SEAR only requires hashes, a key or a nonce added in RREQ, RREP and RERR routing messages. Typically, the size of the hash fields and MACs are much smaller than the size of a signature, leading to SEAR requiring less communication overhead than either ARAN or SAODV.

Message Forgery: Both SAODV and ARAN append digital signatures to each routing message to prevent message forgery. SEAR uses the inherent properties of the AODV routing algorithm to assure that, even if an adversary attempts to forge old or less optimal routes, these routes will be suppressed and not introduce any benefit to an adversary.

Increase Sequence Number: SAODV stops the adversaries from increasing the sequence numbers by adding digital signatures to each routing message. While SEAR achieves the same goal by using efficient one-way functions.

Decrease Hop Count: Both of SAODV and SEAR prevents the decrease of the hop count by using one-way functions.

Same Hop Count Fraud: SAODV only prevents the decrease of the hop count, while attackers can still transmit routing messages with the same hop count as the messages they receive. SEAR encodes node identity into sequence numbers and hop counts, hence attackers would have to increase the hop count as they forward the messages. In ARAN, malicious nodes can forward the routing packets without replacing the signature. Malicious nodes can even strip the outer signature and replay these messages as if it were the source.

Signature DoS: Both ARAN and SAODV are based on expensive public key operations, therefore both are susceptible to signature DoS attacks. Since SEAR is based on efficient one-way hash functions during the main operation of the protocol, it is not vulnerable to signature denial of service attacks after the initial bootstrap phase.

In summary, SEAR's design and use of symmetric cryptographic operations gives significantly less computation and communication overhead with better security than SAODV or ARAN.

5.2 Performance Analysis

We used $ns2$ to study the performance efficiency of SEAR and SAODV when there is no attacker. We assume that all nodes were loosely time synchronized with each other with fixed synchronization errors. Further, we assume that each node had securely distributed the authenticator commitments to every other node in the network and that TESLA key commitments were established with neighbors.

In the simulations, nodes moved following the random waypoint mobility model with a maximum speed $20m/s$. The simulation space was a rectangular region with a size of $1500m \times 300m$ with 50 nodes. The maximum end to end network delay was $0.1s$. The communication range for each node was set to $250m$. There were a total 15 pairs of communicating nodes, with each source sending out constant bit rate (CBR) traffic with packet sizes of 64 bytes at a rate of 4 packets/second. The link bandwidth was set to $1Mbps$. The hash size, MAC size and key size were set to 80 bits, while the signature size was set to 1024 bits. The TESLA time interval was set to $1s$, and the synchronization error was set to $0.1s$. The time to generate a signature was set to $10ms$ and the time to verify a signature was set to $1ms$. We omitted the time needed to compute hashes in the simulations.

We studied the performance of SEAR for both no identity encoded and with identity encoded versions. Each node had a unique hash which corresponded to its identity. Further, we compared the performance of SEAR with SAODV under the same network topology and simulation parameters. In order to maximize the advantages of SAODV, we performed the simulations for both cache-enabled and cache-disabled versions of SAODV. We evaluated the performance of SEAR by comparing the following metrics to those of AODV and SAODV:Packet delivery ratio, routing overhead (in terms of number of packets), routing overhead (in terms of number of bytes), and packet delivery delay.

The simulation results are shown in Fig. 2, which is based on an average over 60 runs of different movement files for each pause time. The 95% confidence intervals for the metrics are plotted as error bars. The packet delivery ratio of SEAR with either identity encoded or without identity encoded degrades at most 1% for all pause times, which illustrates that SEAR performs better than both versions of SAODV. The packet delivery ratio for SAODV with

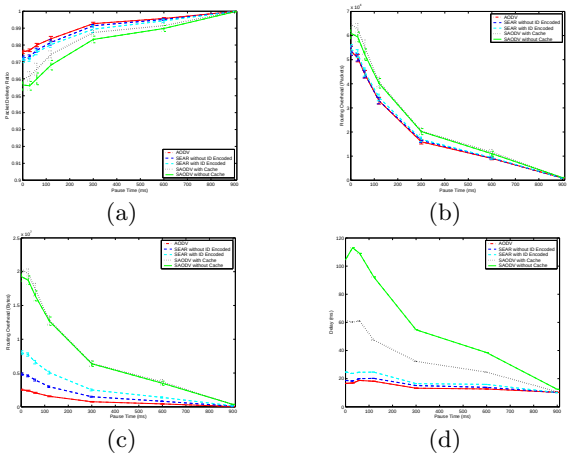

Figure 2: Performance comparison for AODV, SEAR and SAODV. (a) PDR, (b) Overhead in packets, (c) Overhead in bytes, and (d) delivery delay.

cache-enabled is better than that for SAODV without cache. The routing overhead in terms of the number of packets for SEAR is at the same level as AODV. For SEAR, the routing overhead in terms of bytes transmitted when identity is not encoded is about twice the amount of baseline AODV. The routing overhead in terms of number of bytes when identity is encoded is higher than without identity encoded. We also observed that the amount of bytes needed for routing overhead was roughly 4 times larger for SAODV than SEAR. For SEAR, the average packet delivery delay is slightly increased due to the increased communication overhead, while the increase in delay for SAODV is roughly 3 times with cache enabled and 5 times without cache support. Overall, SEAR outperforms both SAODV versions in all of the performance metrics that we examined.

6. CONCLUSION

In this paper, we have proposed SEAR protocol for wireless ad hoc networks. SEAR is mainly based on symmetric cryptography, while asymmetric cryptography is only used for the initial keys distribution. We have shown through both theoretical examination and simulations that SEAR is less vulnerable to routing attacks when compared to existing secure AODV protocols.

7. REFERENCES

[1] Y. Hu, D. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, I:175–192, 2003.

[2] Y. Hu, A. Perrig, and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, pages 12–23, New York, NY, USA, 2002. ACM Press.

[3] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, 2003. IETF RFC 3561.

[4] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[5] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *Cryptobytes*, 5:2–13, 2002.

[6] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 78–87, 2002.

[7] M.G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of ACM Workshop on Wireless Security (WiSe)*, pages 1–10, 2002.