A Secure Reactive Ad Hoc Routing Protocol

Zhenjiang Li, J.J. Garcia-Luna-Aceves
Department of Computer Engineering, University of California, Santa Cruz
Santa Cruz, CA 95064, U.S.A.
Email: {zhjli@soe.ucsc.edu }

Abstract

A light-weight approach for securing On-demand routing protocols for Ad hoc network - SOARP, is presented. Pairwise shared keys between pairs of mobile nodes and hash values keyed with them (K-MAC: Keyed Message Authentication Code) are employed to authenticate routing messages and the validity of the path selected, which makes our approach computationally efficient compared with prior approaches based on digital signatures. Security measures for preventing, detecting and responding to attacks are also presented to secure the forwarding of data packets in addition to the signaling messages of the routing protocol. Security analysis and simulations show that our approach can be used in Controlled Ad-hoc networks, and effectively thwarts many attacks to the routing process caused by malicious or compromised nodes.

Keywords: Ad-hoc Network, Self-Certified Key, Controlled and Free Ad-hoc Environment

I. Introduction

Mobile Ad-hoc network is a group of wireless nodes that are deployed on-demand as a multi-hop packet radio network without the aid of pre-established infrastructure. Nodes in Ad-hoc network are assumed to be willing to route messages for other nodes, such that every node has the responsibility of a router as well as a common user of the network. Ad-hoc network can be rapidly deployed in critical scenarios such as battle fields and rescue missions of disasters because a prior infrastructure is not necessitated, which makes Ad-hoc network a promising technology for new applications of wireless networks.

However, most researches on Ad-hoc network by far focus only on highly efficient channel utilization and data forwarding algorithms, and relatively less work have been done in the field of securing daily operations of Ad-hoc network. The purpose of this paper is of designing a generic mechanism to secure On-demand (Reactive) routing protocols for Ad-hoc network, which is easy to deploy at startup stage, light-weighted in daily routing operations, but powerful enough to thwart many threats to Ad-hoc routing protocols.

This paper is organized as follows. Section 2 presents the security models of attackers and application environments for Ad-hoc networks. Section 3 is an overview of on-demand routing and possible threats to Ad-hoc routing protocols. Section 4 is the mechanisms we developed to secure On-demand routing in Ad-hoc network. Section 5 is the security analysis, and performance evaluation by simulations is presented in Section 6. Section 7 are discussions about related works and concluding remarks are presented in Section 8.

II. Security Model

A. Application Environment Model

We classify the application environments of Ad-hoc network into two groups according to if the participants (nodes) of network are under the control of a centralized administrative authority.

In **Controlled Ad-hoc Environment**,the Ad-hoc network is under the control of a certain central authority, only authorized nodes can join the network and a prior negotiation or initialization are required before actual deployment of the network. The beforehand negotiation with central authority might involve parameter initialization, or cryptographic keys distribution etc. The typical cases of controlled Ad-hoc environments are critical scenarios such as disaster rescue, battle field etc. In **Free Ad-hoc Environment**, nodes can join and leave Ad-hoc network at will, and there is no restriction for them to negotiate with a certain central administrative authority before they are allowed to take advantage of the network resources. No central administration does not mean no beforehand negotiation for necessary cryptographic keys, which any security measure based on cryptography cannot avoid, but achieve it somehow without the control of the central administration. Free Ad-hoc environments are usually civilian applications, such as participants of temporary conference in a meeting room.

B. Attacker Model

The power of an attacker is determined by his knowledge mastered at the time when he launches the attack, and the complexity of an attack is often proportional to the number of attackers involved. **Malicious Node(MN)** is not legal node and does not have necessary cryptographic keys in order to participate in normal operating of routing process; **Compromised Node(CN)** is legal node but has been taken over by attackers, so attackers possibly have access to the cryptographic keys owned by this node. Because of possible access to related keys, compromised node can launch not only the same attacks as those done by malicious nodes, but is able to originate more complicated attacks 'legally' in a hidden manner, which are difficult to detect; **Group of Adverse Nodes(GAN)** always work collusively and the attacks launched by them are far more delicate than those originated by only one attacker. GAN nodes might include malicious nodes, compromised nodes or nodes with more powerful capabilities than other nodes in network, such as longer range transmission antenna. But they need an appropriative channel to coordinate their actions as to attack collaboratively.

III. On-demand Routing In Ad-hoc Network

A. Overview of On-demand routing

In On-demand (or Reactive) Ad-hoc routing protocols, routes exist only when necessary. For example, if node S(Source) want to communicate with node D(destination) but no route found in local routing table, A route query message $RREQ$ will be generated by S and broadcast into the network, further forwarded by any intermediate node receiving it, until it arrives the destination D. Intermediate nodes also setup corresponding reverse forwarding information when they are forwarding a $RREQ$. When destination node finally receives $RREQ$, it will unicast back a "Route Reply" message $RREP$ to the source S via the reverse path recorded in intermediate nodes. At the same time when intermediate nodes are reversely forwarding $RREP$, they also setup the corresponding data forwarding entry for later packets targeting at the destination D. The source node may receive multiple $RREP$s and will choose one neighbor node as the next hop for the destination according to some specific metrics ,such as to choose the route with minimum hops to destination.

B. Threats to Ad-hoc routing protocols

The attacks to Ad-hoc routing protocol could be classified according to the originator of attackers. **External Attacks** are originated by malicious nodes ,who lack necessary keys as to participate in normal routing operation. Common external threats include impersonation(spoofing), meaning an attacker attempts to impersonate another one; modification, meaning entries in routing messages have been modified; fabrication, meaning attackers fabricate fake routing messages and interfere with normal routing operation by inserting them into the network etc. **Internal Attacks** are originated by compromised nodes, who might have necessary keys to participate in routing operation. Internal attacks are formally similar to external attacks, but they are far more difficult to detect because attackers have access to valid keys. **Complicated Attacks** are often launched by GAN nodes collusively. "Worm-hole" attack is a typical example, where two GAN nodes tunnel a packet from one point of network to another point far away enough, in hope that any receiver of this packet would believe the sender of this packet is in his close neighborhood. Another instance is D-DOS (Distributed Deny Of Service) launched by a group of nodes. D-DOS are more likely to interrupt the communication of network than DOS done by a single attacker because the power and resources owed by a single node is usually much more limited than that of a group of attackers.

IV. Methodology

A. Assumptions

We assume that each pair of users (Node N_i and node N_j)in the network shares a pairwise secret key $K_{i,j}$, which is used for the authentication of route discovered during route discovery process. We will present the key distribution scheme in next subsection, which is based on the Self-Certified Key (SCK) [9]. The underlying wireless link is bidirectional, meaning if node S can send packet to node D, then S can also overhear the transmission of node D.

B. Self-certified public key cryptosystem [9]

In asymmetric public key cryptosystem, each user has a pair of secret and public keys for encryption/decryption, digital signature etc. Before a public key could be used, the binding between this public key and its owner must be authenticated, thus to avoid masquerade attacks. There are two ways of ensuring the authenticity of a public key: explicit verification and implicit verification. In explicit verification, a trusted CA (certificate authority) will sign a certificate which binds a public key and the ID of its owner, then any user can verify the certificate explicitly given the public key of the CA. In implicit verification, the authenticity of a public key is verified when it is used for decryption, signature verification, and other cryptographic operations. For example, a successful verification of a signature means the public key does match the secret key used to sign this signature.

Self-certified key (SCK) system follows the track of implicit verification. In SCK, the public key of a user U is not generated and signed by the CA explicitly. Everyone, having the knowledge of the ID and a published piece of data of user U, will be able to derive the public key of U, given the public key of CA. The following advantageous we will exploit are those features of SCK which are very convenient and efficient for our security mechanisms to secure reactive Ad-hoc routing protocols.

- Given N users in network, assuming their IDs are known to all users, in order to distribute their public keys, N credentials (Called guarantees) are to be distributed, instead of N traditional certificates. The advantage is, unlike certificate based asymmetric cryptosystem, these N guarantees can be published and need not to be certified (signed) by any trusted authority, meaning we can distribute the public keys without the aid of on-line central administration like CA(Access to CA is only required at the very initial stage of key generation, as described in shortly later). The binding of a guarantee and its owner (More specific, the corresponding secret key)will be authenticated when cryptographic operations (with keys derived from the guarantee) are performed.
- Given N guarantees are already distributed to all users of the network, and assuming the public key of CA is known to everyone, any two users could compute a pairwise shared key known only to these two users in a non-interactive manner, meaning no further negotiation message needed for key agreement between these two users, which is a big advantage over traditional interactive key exchange methods such as Diffie-Hellman key exchanging protocol. Then these pairwise shared keys could be used to encrypt data between users by conventional symmetric key encryption algorithms, or to compute K-MAC (Keyed Message Authentication Code) by performing one-way hash operation etc. According to [9], non-interactive key progression and agreement in SCK consist of following stages.

TABLE I

KEY AGREEMENT BETWEEN ALICE AND BOB

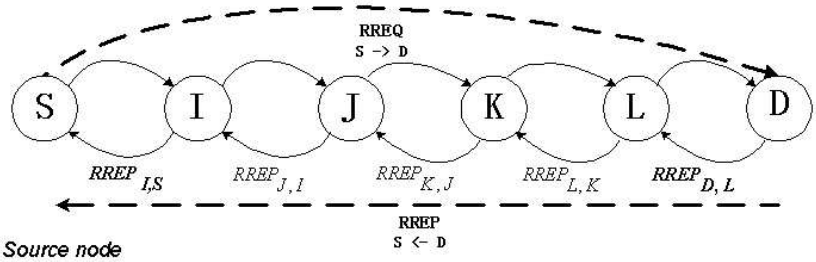
User Alice:	**User Bob:**
$x_{A,t} = x_{A,0} \cdot h(ID_A, r_{A,t}) + k_{A,t}$	$x_{B,t} = x_{B,0} \cdot h(ID_B, r_{B,t}) + k_{B,t}$
$y_{B,t} = y_{B,0}^{h(ID_B, r_{B,t})} \cdot r_{B,t} \ (mod\ p)$	$y_{A,t} = y_{A,0}^{h(ID_A, r_{A,t}} \cdot r_{A,t} \ (mod\ p)$
$K_{A,t} = y_{B,t}^{x_{A,t}} \ (mod\ p)$	$K_{B,t} = y_{A,t}^{x_{B,t}} \ (mod\ p)$
$K_t = h(K_{A,t})$	$K_t = h(K_{B,t})$

1) **Initialization** A certificate authority Z is assumed to be existed, and Z chooses large primes p, q with $q|(p-1)$ (i.e., q is a prime factor of $p-1$), a random number $k_A \in_R Z_q^*$, where Z_q^* is a multiplicative subgroup with order q and generator α, then Z generates its secret-public key pair (x_Z, y_Z) ,and we assume that the public key y_Z is known to everyone in network. To issue the secret key for user Alice, Z computes the signature parameter $r_A = \alpha^{k_A} \ (mod\ p)$ and $s_A = x_Z \cdot h(ID_A, r_A) + k_A \ (mod\ q)$, where $h()$ is a collision-free hash function. Then Alice publishes the parameter r_A, also called guarantee, together with her identifier ID_A and keep the $x_A = s_A$ as her secret key. Her corresponding public key could be computed by everyone who knows the y_Z, ID_A and r_A as
$$y_A = y_Z^{h(ID_A, r_A)} \cdot r_A \ (mod\ p)$$
We denote this initial basic key pair as $(x_{A,0}, y_{A,0})$
2) **User-controlled key progression** Alice can switch her keys either after a fixed time interval of length l or after an arbitrary time period. Now we assume the first setting, where Alice will use key pair $(x_{A,t}, y_{A,t})$ in time interval $[t \cdot l, (t+1) \cdot l]$. Alice then chooses n random pairs $\{k_{A,i} \in_R Z_q^* \ , \ r_{A,i} = \alpha^{k_{A,i}} \ (mod\ p)\}$ where $1 \leq i \leq n$, then the key progression is defined by
$$x_{A,t} = x_{A,0} \cdot h(ID_A, r_{A,t}) + k_{A,t} \ (mod\ q)$$
And the corresponding public key is given by
$$y_{A,t} = y_{A,0}^{h(ID_A, r_{A,t})} \cdot r_{A,t} \ (mod\ p)$$
3) **Agreement of the pairwise shared keys**
The non-interactive algorithm of key agreement between two users are presented in Table I, and the pairwise shared key obtained by Alice and Bob is equal because
$h(K_{A,t}) = h(y_{B,t}^{x_{A,t}} \ (mod\ p)) = h(\alpha^{x_{A,t} x_{B,t}} \ (mod\ p)) = h(y_{A,t}^{x_{B,t}} \ (mod\ p)) = h(K_{B,t})$

C. Approach to secure Ad-hoc On-demand routing protocols

According to the generic operating of On-demand routing in Ad-hoc network, our protocol is divided into several phases , which are corresponding to different On-demand routing stages. A simple example and flow of routing messages are shown in Figure 1 for illustration of following descriptions.

- **Route Query Initialization**
When node S(Source) want to communicate with node D(Destination) but no route entry for D existed in local routing table, S will generate a route query message $RREQ$ and broadcast it into the network. The $RREQ$ has following format:
$$RREQ = \{RREQ, S, D, QNum, SMAC_{s,d}\}, where$$
$RREQ$ is the type identifier of route query message, S, D are the IDs of source and destination node respectively, and $QNum$ is a randomly generated route query number. Then the IDs of nodes S, D and random query number $QNum$ uniquely identify current round of route discovery for specific destination D. The keyed-MAC (Message Authentication Code) value $SMAC_{s,d} = Hash(RREQ, S, D, QNum, Key_{s,d})$ will serve as a signature of S for destination to verify the integrity of this $RREQ$ and it is really generated by source S. We use $Key_{i,j}$ to represent the pairwise shared key between nodes N_i and N_j, which is afore-hand computed according to the non-interactive key exchange algorithm described in previous subsection, obviously $Key_{i,j} = Key_{j,i}$.
- **Route Query Forwarding**
Any intermediate node receiving $RREQ$ will further forward it if this $RREQ$ has never been received (with same S, D and $QNum$), or the $RREQ$ will be dropped. At the same time, reverse forwarding info towards the source node will also be setup in the intermediate nodes.
- **Checking $RREQ$ At Destination D**
D will not further forward $RREQ$ when receiving $RREQ$ because he is the destination. First, D will check to see if this $RREQ$ has been received from a same neighbor before according to the query number carried; Second, D will authenticate if this $RREQ$ is really originated from S and no content has been modified during transmission by evaluating $Hash(RREQ, S, D, QNum, Key_{d,s}) = SMAC_{s,d}$. If both checks are successful, D will initialize a route reply process.



S: Source node
D: Destinaion node
{I,J,K,L}: Intermediate nodes
***SMAC*: This hash value will be generated and used only once**
***MMAC*:This hash value might be used as input of another hash operation**

$$S \rightarrow D: RREQ = \{RREQ, S, D, QNum, SMAC_{S,D}\}, SMAC_{S,D} = Hash(RREQ, S, D, QNum, Key_{S,D})$$

$D \rightarrow S$: *RREP will follow the reverse path setup by intermediate nodes*

$$D \rightarrow L: RREP_{D,L} = \{RREP, D, L, RNum, 0, SMAC_{D,L}, \{Null\}, MMAC_{D,S}\}$$
$$MMAC_{D,S} = Hash(0, CORE, \{Null\}, Key_{D,S})$$
$$CORE = Hash(D, S, RNum, Key_{D,S}), RNum = QNum + 1$$
$$SMAC_{D,L} = Hash(RREP, D, L, RNum, 0, \{Null\}, Key_{D,L})$$

$$L \rightarrow K: RREP_{L,K} = \{RREP, L, K, RNum, 1, SMAC_{L,K}, \{Null, D\}, MMAC_{L,S}\}$$
$$MMAC_{L,S} = Hash(1, MMAC_{D,S}, \{Null, D\}, Key_{L,S})$$
$$SMAC_{L,K} = Hash(RREP, L, K, RNum, 1, \{Null, D\}, Key_{L,K})$$

$$K \rightarrow J: RREP_{K,J} = \{RREP, K, J, RNum, 2, SMAC_{K,J}, \{Null, D, L\}, MMAC_{K,S}\}$$
$$MMAC_{K,S} = Hash(2, MMAC_{L,S}, \{Null, D, L\}, Key_{K,S})$$
$$SMAC_{K,J} = Hash(RREP, K, J, RNum, 2, \{Null, D, L\}, Key_{K,J})$$

$$J \rightarrow I: RREP_{J,I} = \{RREP, J, I, RNum, 3, SMAC_{J,I}, \{Null, D, L, K\}, MMAC_{J,S}\}$$
$$MMAC_{J,S} = Hash(3, MMAC_{K,S}, \{Null, D, L, K\}, Key_{J,S})$$
$$SMAC_{J,,I} = Hash(RREP, J, I, RNum, 3, \{Null, D, L, K\}, Key_{J,I})$$

$$I \rightarrow S: RREP_{I,S} = \{RREP, I, S, RNum, 4, SMAC_{I,S}, \{Null, D, L, K, J\}, MMAC_{I,S}\}$$
$$MMAC_{I,S} = Hash(4, MMAC_{J,S}, \{Null, D, L, K, J\}, Key_{I,S})$$
$$SMAC_{I,S} = Hash(RREP, I, S, RNum, 4, \{Null, D, L, K, J\}, Key_{I,S})$$

Fig. 1. Illustration of secure routing

- **Route Reply Initialization**
  D generates a route reply message $RREP$ according to the $RREQ$ received, specifically

  $$RREP = \{RREP, D, N_1, RNum, HOPS, SMAC_{d,1}, \{NodeList\}, MMAC_{d,s}\},\ where$$

  $RREP$ is the type identifier of route reply message and N_1 is the upstream neighbor, from which the $RREQ$ was received; route reply number $RNum$ is equal to $QNum$ plus one; $HOPS$ represents the hop-count to destination, here $HOPS = 0$ because D is the destination itself; $SMAC_{d,1} = Hash(RREP, D, N_1, RNum, HOPS, \{NodeList\}, Key_{d,1})$ is the K-MAC for the purpose of neighbor-by-neighbor authentication between D and N_1; the $\{NodeList\}$ field will record all the intermediate nodes gone through by the $RREP$, and now $\{NodeList\} = \{Null\}$; the other k-MAC $MMAC_{d,s} = Hash(HOPS, CORE, \{NodeList\}, Key_{d,s})$ will be further processed by intermediate nodes and be used by source S to authenticate the validity of the path reported by $RREP$; $CORE = Hash(D, S, RNum, Key_{d,s})$ serves as a signature of D for source S to authenticate that this $RREP$ is really generated by D

  [Node Numbering Rule:] $\{NodeList\}$ actually represents a path from S to D. For clarity, they are numbered increasingly from D to S. For example, a $\{NodeList\}$ could looks like $\{D, N_1, N_2, N_3, ...N_{hops}\}$.
- **Reverse Route Reply Forwarding**
  Any intermediate node N_i in the reverse route will authenticate, process and further forward received $RREP$ to its upstream node N_{i+1}, which was recorded in its reverse forwarding table. Firstly, N_i will check to see if this $RREP$ has been received before according to the route reply number carried; Secondly, checking if this $RREP$ is truly forwarded by the downstream node N_{i-1} and its contents are not modified by evaluating

$SMAC_{i-1,i} = Hash(RREP, N_{i-1}, N_i, RNum_{rep}, HOPS_{rep}, \{NodeList\}_{rep}, Key_{i,i-1})$, $where$
$SMAC_{i-1,i}$, $RNum_{rep}$, $HOPS_{rep}$ and $\{NodeList\}_{rep}$ are the corresponding fields in received $RREP$ message.
If both verifications are passed, N_i will update and further back-forward $RREP$ to upstream node N_{i+1} as:
$RREP = \{RREP, N_i, N_{i+1}, RNum, HOPS, SMAC_{i,i+1}, \{NodeList\}, MMAC_{i,s}\}, where$
$HOPS = HOPS_{rep} + 1$, and $SMAC_{i,i+1} = Hash(RREP, N_i, N_{i+1}, RNum, HOPS, \{NodeList\}, Key_{i,i+1})$;
$\{NodeList\} = \{\{NodeList\}_{rep}, N_{i-1}\}$, meaning N_i will append the ID of the downstream neighbor node, from which it received the $RREP$, into the new $\{NodeList\}$; $MMAC_{i,s} = Hash(HOPS, MMAC_{i-1,s}, \{NodeList\}, Key_{i,s})$, where $MMAC_{i-1,s}$ is copied from the $MMAC$ field of received $RREP$

- **Checking** $RREP$ **At Source** S
  The verification processing will be strait-forward if presented in "C-style" statements as shown in Figure 2. Basically, S

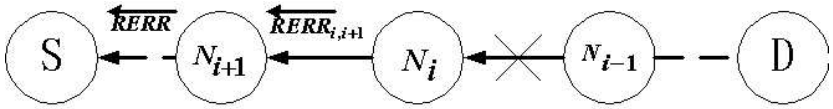
```
if [ RNum = QNum + 1 ]
  if [ SMAC_{hops,s} = Hash(RREP,N_{hops},S,RNum,HOPS, {D,N_1,...,N_{hops-1}},Key_{s,hops}) ] {
    CORE = Hash(D,S,QNum+1,Key_{s,d});
    {NodeList} = {Null}
    MMAC_{temp} = Hash(0,CORE,{Nul},Key_{s,d});
    for ( j=0; j < HOPS; j++ ){
        {NodeList} = {NodeList,N_j}; /* N_j = D if j=0 */
        MMAC_{temp} = Hash(j,MMAC_{temp},{NodeList},Key_{s,j}); }
    if [ MMAC_{hops,s} = MMAC_{temp} ]
       Accept RREP and use {NodeList,N_{hops}} as the route for D;
    else Drop RREP; endif
  else Drop RREP; endif }
else Drop RREP; endif
```

Fig. 2. Checking $RREP$ at source S

repeats the same computations done by all the intermediate nodes from D to S (But with the keys owned by himself), which were recorded in the $\{NodeList\}$ of $RREP$, and the number of hash operations need to perform is equal to the $HOPS$ reported by this $RREP$.
If above verification is successful , S could believe that the $RREQ$ has reached destination D, every node listed in $\{NodeList\}$ actually participated in the reversal forwarding of $RREQ$ and these intermediate nodes do provide a path from S to D.

D. Route Maintenance

When node N_i on the path selected by source S to destination D finds the downstream link/route to destination D is broken, N_i will generate a route error report massage $RERR$ and unicast it back to the source S via the reverse path setup before, which might invoke S to start a new round of route discovery. Before accepting the $RERR$ message, we must be able to ensure: the node generating the $RERR$ message must be on the path to the destination; the node reporting link failure should still be there when he reported the error. The process of sending back a $RERR$ message from node N_i is almost the same as that when node N_i originates a route reply to the source S. We only describe the main differences here and demonstrate more details in Figure 3.



$N_i \rightarrow N_{i+1}$: $RERR_{i,i+1} = \{RERR, N_i, N_{i+1}, RNum, 0, SMAC_{i,i+1}, \{Null\}, MMAC_{i,S}\}$
$MMAC_{i,S} = Hash(0, CORE, \{Null\}, Key_{i,S})$
$CORE = Hash(...Hash(N_i, S, RNum, Key_{i,S}))$, i times of hash() operations
$SMAC_{i,i+1} = Hash(RERR, N_i, N_{i+1}, RNum, 0, \{Null\}, Key_{i,i+1})$

Fig. 3. N_i generate $RERR$ if downstream link/route is broken

The $RERR$ message has the similar format to $RREP$, except that the message type identifier and the computation of $CORE$, which is computed as $CORE = Hash(...Hash(N_i, S, RNum, Key_{i,S}))$, here i times Hash operations and i is equal to the hops from node N_i to destination D. Nodes on the path will only process and back-forward a $RERR$ received from its

downstream node to destination D, which ensure the node N_i is still on the path to D when he reported the link failure. When source node S finally received the $RERR$, he will perform the similar authentication procedure to that of a $RREP$, the only difference part is the computation of $CORE$, which is shown as following, $CORE = Hash(...Hash(N_i, S, RNum, Key_{S,i}))$, here $(HOPS_{S \to D} - HOPS_{Reported\ by\ RERR} - 1)$ times hash computations will be performed.

After authenticating the $RERR$, source S can initiate a new round of route query for destination D, if further communication with D is still needed.

E. Secure Packet Delivery

So far we have discussed security measures which are sufficient enough for the source to acquire route info for a destination securely, but we still have no confidence in succeeding data delivery process. In order to achieve reliable packet delivering, neighbor-by-neighbor downstream monitoring and responding mechanism are introduced into our protocol, which is shown in Figure 4. Any upstream neighbor on the path from source S to destination D will monitor the transmission of its downstream

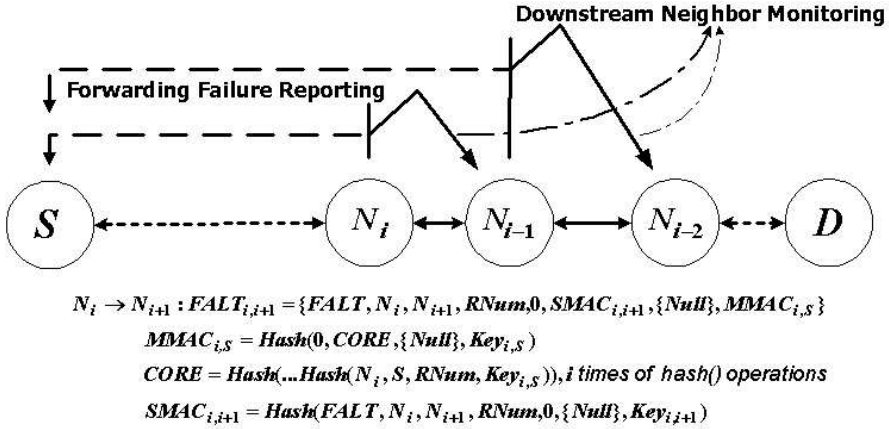


Fig. 4. Illustration of Securing Packet Delivery

neighbor node, which could be done because we assume the wireless link is bidirectional. If the monitoring node find that the downstream neighbor failed to further forward data packets to destination D (Say below a pre-established threshold of packet forwarding ratio), then the monitoring node will initialize a forwarding failure alert process which is very similar to the error report process of link failure, except a different message type identifier $FALT$. In this way, the source node S can have timely notification if any node failed to deliver data packets and S could also know where the problem happened. Once receiving a forward failure alert, the source might re-initialize a new round of route discovery process to discover another route which avoids the suspicious intermediate node. The verification of $FALT$ message is same as $RERR$ message at the source node S given $FALT$ follows the same format and back-forwarding process as that of $RERR$, which are illustrated in Figure 4. We admit that the monitoring mechanism used here are still naive, but we believe better detection and response techniques will be developed and they are also part of our future work.

V. Security Analysis

For external attacks, because a malicious node MN has no necessary keys shared with other nodes as to participate in normal routing operation, any modification, fabrication ,replay and impersonation (spoofing) will be detected by either intermediate nodes locally or by source node at last. For internal attacks, a compromised node CN cannot impersonate another node N_i because CN lacks of the secret key know only to N_i, therefore, the compromised node cannot derive correct pairwise shared keys between N_i and other nodes. CN could modify fields in $RREP$, such as hop-count, but it will be detected by source node S because hash values are correlated with the chain of whole intermediate nodes ($MMAC$ is keyed with the shared keys between intermediate nodes and the source) and also related to the hop-count from S to D. Route loops could be detected at the same time when S is checking the validity of the path because $\{NodeList\}$ is provided. Due to the use of randomly generated route query number, simple replay of $RREQ$ and $RREP$ will be easily detected by either intermediate nodes or source/destination node. As mentioned earlier, the attacks launched by a group of adverse nodes (GAN) could be far more complicated than attacks originated by a single adversary because of the information and power they control. Also we cannot enumerate all the possible attack patterns generated by GANs, thus approaches based only on prevention might lose its effect when the threats confronting are not those they are designed to handle. The approach we choose is a combination of preventing, monitoring and responding, and try to maintain a reliable communication system at acceptable cost. On one hand, the K-MAC based secure routing measures built in SOARP could be able to detect and thwart most common attacks aimed at Ad-hoc routing, as explained before; on the other hand, neighbor-by-neighbor monitoring and necessary forwarding failure alert will find packet delivery failure timely and invoke source to initialize another round of route discovery for future data delivery, no matter what kind of attack incurred that failure. Simulation results have shown that, which will be presented

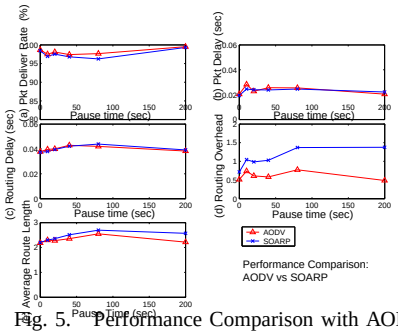


Fig. 5. Performance Comparison with AODV

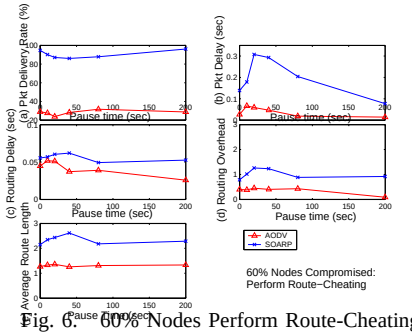


Fig. 6. 60% Nodes Perform Route-Cheating

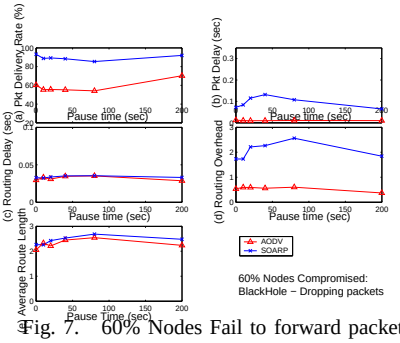


Fig. 7. 60% Nodes Fail to forward packets

shortly, even these simple monitoring and responding methods could achieve satisfactory performance even 60% nodes in network are compromised.

VI. Evaluation

In order to evaluate the performance and effectiveness of SOARP, we carry out simulations in network simulator NS-2 [1]. In our simulation configuration, the field size is 750m x 500m with 30 nodes moving around according to the random way-point model. Each node is positioned at a random place at the beginning of simulation, and pauses for a certain interval called pause time; then it will choose a new place and move there at a velocity uniformly distributed between 0 and 20m/s; when it get there, it will pause again and choose a new position to which to move, so on and so forth. The traffic pattern we use is 10 randomly chosen source-destination sessions, each of them is a Constant Bit Rate (CBR) flow at a rate of 4 packets per second, and 512 bytes per packet in size; the radio range of underlying wireless link is 250m. The hash implementation we use is MD5 from RSAREF library, which is public available for research purpose and proven to be strong enough. In another words, we model the computational cost and according delay incurred by the authentication processing in SOARP.

The metrics we used to measure the performance are: **Packet Delivery Ratio** the total number of CBR packets received, over the total number of CBR packets originated, over all the mobile nodes in the network; **End2End Packet Delay** the average elapsed time between a CBR packet is passing to the routing layer and that packet is received at the destination node; **Route Discovery Delay** the average time it takes for a node to find a route to a destination; **Normalized Routing Overhead** the total routing messages transmitted/forwarded over the total number of CBR packets received, over all mobile nodes; **Average Route Length** the average route length (hop counts) of the paths discovered by nodes to transmit data packet, over all mobile nodes in network.

A. Performance without attackers

First we will see the performance difference between SOARP and Ad-hoc On-demand Distance Vector(AODV) Routing protocol [8], which is one of the leading Ad-hoc routing protocols, if there is no attackers in the network. The results are shown in Figure 5 as a function of pause time. Each figure represents the average over runs of 10 random movement patterns at each pause time, and the same patterns are used for both SOARP and AODV. The simulation time we choose is 200 seconds, which means the nodes are stationary when the pause time is 200 second.

As shown in the (a), (b), (c) and (e) of Figure 5, the packet delivery ratio, packet delivery delay, average time to find a route and average route length are very close for both AODV and SOARP, which means SOARP is almost as efficient as AODV in routing discovery and data delivery, and K-MAC (hash computation) based security measures integrated in SOARP do not incur much delay in route discovery and data delivery. SOARP does introduce higher route overhead than AODV, as shown in (d) of Figure 5, which is mainly due to the neighbor by neighbor forwarding monitoring alert messages, because broken of link will also result in failure of packets forwarding, which will trigger the upstream node to generate forwarding failure alerts to source node.

B. Performance with attackers

To study the effectiveness of SOARP, we also design and carry out simulations when partial of the nodes in network are compromised. Figure 6 shows the same evaluation metrics when 60 percent of the nodes are compromised and these compromised nodes will fabricate fake route replies to any route query by claiming they are “zero” hop to the destination node, and drop all the succeeding data packets passing through them. The purpose of this simulation is to study the effectiveness of secure routing measures built in SOARP.

We can notice that the packet delivery ratio of AODV decreases dramatically, (a) of Figure 6 , because most of the packets will be sent to the wrong routes and dropped by compromised nodes. When routing with AODV, the source node might be misled by incorrect routing replies and choose a route with a length much shorter, (e) of Figure 6, than when there is no compromised node, (e) of Figure 5; Because of that, the routing overhead is relatively lower, (d) of Figure 6, and average time to find a route is also shortened, (c) of Figure 6.

On the other side, SOARP still performs well with over 80 percent packet delivery rate at all pause time runs, (a) of Figure 6. The costs of this are longer average time to find a route to destination, end-2-end packet delay and higher routing overhead, as shown in (b) (c) (d) of Figure 6 respectively; the average route length found by SOARP is longer than that of AODV because SOARP cannot be misled by any route reply from compromised nodes, (e) of Figure 6.

Figure 7 shows the simulation results when 60 percent of the nodes are compromised, but they will follow regular routing operations correctly, and just drop any sequential packet passing through them (Black-Hole Attack). The purpose of this simulation is to study the effectiveness of neighbor by neighbor monitoring and responding mechanisms of SOARP. We can notice that SOARP can still achieve over 80 percent packet delivery ratio at all runs, while AODV only achieve 55 percent, (a) of Figure 7; the price to pay is higher routing overhead because more forwarding failure alerts and routing discovery messages incurred, (d) of Figure 7, and relatively larger packet delivery delay (b) of Figure 7; the average time to find route and average route length are similar to that when there is no compromised node in network, because all nodes still cooperate correctly in route discovery process, and the field size we use is relatively of a small scale, (c) (e) of Figure 7.

VII. Related Work

Recently, many efforts have been made to secure routing protocols specific for Ad hoc network. SAR [12] organizes nodes of a network into a trust hierarchy with different trust levels, nodes on the same level share a session key, and a node can process and further forward messages only if the node itself is on or above a specified level of trust. Authors of [7] use keyed MAC to authenticate route request and reply messages between source and destination. Because authentication is end-2-end based and there is no authentication for intermediate nodes, it might be possible for attackers to impersonate other nodes in the middle of the path, even just temporarily. Ariadne [2] is designed for reactive routing protocol DSR [3] and assumes shared secrets exist between each pair of nodes and a broadcast authentication scheme such as TESLA is required. For Ariadne to work properly, time synchronization among all nodes is assumed, which might be so demanding for Ad-hoc network in some scenarios. ARAN [10] has a similar route discovery process like common reactive Ad-hoc routing protocols and digital signature is used for the authentication of routing messages between source and destination. The work in [6] presents a self-organizing scheme based on threshold signing algorithm which could provide authentication service for all nodes in network and well scale to large size network. Some other works also based on threshold signing can be found in [5], [11], [4]. The protocol proposed in [13] uses information of hop-count together with hash chains to authenticate mutable part in routing messages, and digital signature is introduced to protect the immutable fields of signaling messages, it is computationally expensive because intermediate nodes need to perform a signature verification for every message received.

VIII. Conclusion

In this paper, we present a light-weight protocol of securing Reactive routing in Ad-hoc network. Our protocol is ready to deploy and computationally efficient comparing to digital signature based approaches, because pairwise shared keys between each pair of mobile nodes and hash values keyed with them (K-MAC: Keyed Message Authentication Code) are employed to authenticate routing messages and the validity of the path selected. Combined security measures for preventing, detecting and responding to attacks are used to secure both the delivery of data packets and the signaling messages of the routing protocol.

References

[1] The network simulator - ns-2, http://www.isi.edu/nsnam/ns/.

[2] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Y.-C. Hu, A. Perrig, and D. B. Johnson, Ariadne: A secure on-demand routing protocol for ad hoc networks, in The 8th ACM International Conference on Mobile Computing and Networking*, September 2002.

[3] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[4] Jiejun Kong, Haiyun Luo, Kaixin Xu, Daniel Lihui Gu, Mario Gerla, and Songwu Lu. Adaptive security for multi-layer ad-hoc networks. In *issue in John Wiley InterScience Press journal of Special Issue of Wireless Communications and Mobile Computing*, Aug. 2002.

[5] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *IEEE 9th International Conference on Network Protocols (ICNP'01)*, 2001.

[6] Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang. Self-securing ad hoc wireless networks. In *Seventh IEEE Symposium on Computers and Communications (ISCC'02)*, 2002.

[7] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference(CNDS 2002),San Antonio, TX*, Jan. 2002.

[8] C. Perkins. Ad hoc on demand distance vector (aodv) routing, November 1997.

[9] Holger Petersen and Patrick Horster. Self-certified keys - concepts and applications. In *Proceedings of 3rd Conference of Communications and Multimedia Security, Athens*, September 22-23, 1997.

[10] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Shields. A secure routing protocol for ad hoc networks. In *Proceedings of the 10 Conference on Network Protocols (ICNP)*, 2002.

[11] Hao Yang, Xiaoqiao Meng, and Songwu Lu. Self-organized network layer security in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'02), Atlanta*, September 2002.

[12] Seung Yi, Prasad Naldurg, and Robin Kravets. A security-aware ad hoc routing protocol for wireless networks. In *The 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, 2002.

[13] Manel Guerrero Zapata and N. Asokan. Securing ad-hoc routing protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002), pages 1-10*, September2002.