Global Journal of Computer Science and Technology
Volume 11 Issue 20 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

SEAD-FHC: Secure Efficient Distance Vector Routing with Fixed Hash Chain length

By Prasuna V G, Dr. S. Madhusudhana Verma
Rayalaseema University, Kurnool, Andhra Pradesh

Abstract - Ad hoc networks are highly dynamic routing networks cooperated by a collection of wireless mobile hosts without any assistance of a centralized access point. Secure Efficient Ad hoc Distance Vector (SEAD) is a proactive routing protocol, based on the design of Destination Sequenced Distance Vector routing protocol (DSDV). SEAD provides a robust protocol against attackers trying to create incorrect routing state in the other node. However, the computational cost creating and evaluating hash chain increases if number of hops in routing path increased. In this paper, we propose Secure Efficient Ad hoc Distance Vector with fixed hash chain length in short SEAD-FHC protocol to minimize and stabilize the computational complexity that leads minimization in delay time and maximization in throughput. A series of simulation experiments are conducted to evaluate the performance.

Keywords : Mobile ad hoc networks; Ad hoc network routing; Secure routing; SEAD; Hash chains.

GJCST Classification : E.2, E.3

SEAD-FHC SECURE EFFICIENT DISTANCE VECTOR ROUTING WITH FIXED HASH CHAIN LENGTH

SEAD-FHC: Secure Efficient Distance Vector Routing with Fixed Hash Chain length

Prasuna V G [α], Dr. S. Madhusudhana Verma [Ω]

Abstract - Ad hoc networks are highly dynamic routing networks cooperated by a collection of wireless mobile hosts without any assistance of a centralized access point. Secure Efficient Ad hoc Distance Vector (SEAD) is a proactive routing protocol, based on the design of Destination Sequenced Distance Vector routing protocol (DSDV). SEAD provides a robust protocol against attackers trying to create incorrect routing state in the other node. However, the computational cost creating and evaluating hash chain increases if number of hops in routing path increased. In this paper, we propose Secure Efficient Ad hoc Distance Vector with fixed hash chain length in short SEAD-FHC protocol to minimize and stabilize the computational complexity that leads minimization in delay time and maximization in throughput. A series of simulation experiments are conducted to evaluate the performance.

Keywords : Mobile ad hoc networks; Ad hoc network routing; Secure routing; SEAD; Hash chains.

I. Introduction

Secure Ad Hoc network routing protocols are complex to design, due to the generally highly dynamic nature of an ad hoc network and due to the need to operate efficiently with limited resources, including network bandwidth and the CPU processing capacity, memory, and battery power (energy) of each individual node in the network. Existing insecure ad hoc network routing protocols are often highly optimized to spread new routing information quickly as conditions change, requiring more rapid and often more frequent routing protocol interaction between nodes than is typical in a traditional (e.g., wired and stationary) network. Expensive and cumbersome security mechanisms can delay or prevent such exchanges of routing information, leading to reduced routing effectiveness, and may consume excessive network or node resources, leading to many new opportunities for possible Denial-of-Service attacks through the routing protocol.

Routing protocols for ad hoc networks generally can be divided into two main categories: Periodic protocols and On-demand protocols. In a periodic (or proactive) routing protocol, nodes periodically exchange routing information with other nodes in an attempt to have each node always know a current route to all destinations (e.g.,[22,23,24,25,26, 27,28]). In an on-demand (or reactive) protocol, on the other hand, nodes exchange routing information only when needed, with a node attempting to discover a route to some destination only when it has a packet to send to that destination (e.g., [1,29,30]). In addition, some ad hoc network routing protocols are hybrids of periodic and on-demand mechanisms (e.g., [31]).

Each style of ad hoc network routing protocol has advantages and disadvantages. In this paper, we focus on securing ad hoc network routing using periodic (or proactive) protocols, and in particular, using distance vector routing protocols. Distance vector routing protocols are easy to implement, require relatively little memory or CPU processing capacity compared to other types of routing protocols, and are widely used in networks of moderate size within the (wired) Internet [32,33,34]. A number of proposed periodic ad hoc network routing protocols are based on adapting the basic distance vector routing protocol design for use in mobile wireless ad hoc networks, including PRNET [26], DSDV [28], WRP [27], WIRP [25], and ADV [23]. Distance vector routing has also been used for routing within a zone in the ZRP hybrid ad hoc network routing protocol [31].

Ad-hoc network is a computer network in which the communication links are wireless and the devices on it communicate directly with each other. This allows all wireless devices within range of each other to discover and communicate in a peer-to-peer fashion without involving central access points.

An ad-hoc network tends to feature a small group of devices all in very close proximity to each other. Performance degrades as the number of devices grows, and a large ad-hoc network quickly becomes difficult to manage.

To design an Ad hoc network routing protocol is challenging, and to design a secure one is even more difficult. There are many research focus on how to provide efficient [35, 36] and secure [37, 38] communication in ad hoc networks.

The Secure Efficient Ad hoc Distance Vector (SEAD) [40] protocol uses one-way hash chains to prevent an attacker from forging better metrics or sequence numbers. But SEAD does not prevent an attacker from tampering other fields or from using the learned metric and sequence number to send new routing updates. In this paper, we proposed a new

Author [α] : Assocaite Prof., Department Of MCA, Basaveswara Institute Of Information Technology, Hyderabad, AndhraPradesh, INDIA-500027.
Author [Ω] : Professor & Head, Department of OR & SQC, Rayalaseema University, Kurnool, Andhra Pradesh, India – 518002.

protocol to improve security of SEAD. We also conduct some simulation experiments to evaluate the performance of our proposed protocol.

II. Problem Definition

The problem with many routing protocols for ad hoc networks is that those protocols are vulnerable to security attacks. The attacks can be classified as passive or active attacks. In a passive attack, a malicious node ignores operational requirements of the network. For example, an intermediate node along a route does not forward a packet, or hides routing information. Multiple routes and redundant messaging can alleviate passive attacks.

In an active attack, the malicious node introduces false information, e.g., a false distance vector, a false destination sequence, or a false route request. This confuses routing procedures and degrades network performance. With a false route, the malicious node can intercept and comprise packets.

Misdirecting is another active attack. Here, an intermediate node forwards packets along incorrect paths. This attack affects the source node by directing packets away from the intended destination node.

The AODV protocol uses destination sequence numbers to indicate how recently the routing information was generated. When multiple routes are available, the source node always selects a route associated with a largest destination sequence number.

A malicious node can fabricate a false large destination sequence number to attract traffic. Even worse, a deceived node can propagate, in good faith, a false route to other nodes to exacerbate the impact of the attack. In this case, the attacker can maliciously attract and discard data traffic.

A malicious node can also consume a large amount of the network bandwidth by broadcasting fictitious destination addresses to which no node can reply. This delays other traffic and can cause packets to be dropped, lowering overall network performance.

III. Related Work

There are known techniques for minimizing 'Byzantine' failures caused by nodes that through malice or malfunction exhibit arbitrary behavior such as corrupting, forging, and delaying routing messages. A routing protocol is said to be Byzantine robust when it delivers any packet from a source node to a destination as long as there is at least one valid route [3]. However, the complexity of that protocol makes it unsuitable for ad hoc networks.

Papadimitrators et al [4] described a secure routing protocol (SRP) that prevents impersonation and replay attacks for on-demand routing. The protocol disables route caching and provides end-to-end authentication with an HMAC primitive [5]. However, that protocol cannot prevent vicious request flooding because there is no mechanism for authenticating source and intermediate nodes.

Dahill et al[6] introduced another technique uses hop-by-hop authentication. Every node is required to sign and authenticate every message. That increases processing requirements and the size of messages.

Zapata [7] introduced another technique requires that each node has access to a certified public key of all network nodes to validate all routing packets. The originator of a message appends an RSA signature, and a last element of a hash chain, i.e., a result of n consecutive hash calculations on a random number[8, 9]. As the message traverses the network, intermediate nodes can validate cryptographically the signature and the hash value, generate a k^{th} element of the hash chain, with k being the number of traversed hops, and add the hash chain to the message [10].

However, public-key cryptography imposes a high processing overhead on the nodes and may be unrealistic for practical low-cost, ad hoc networks of low-complexity devices, such as sensors. Hash chaining requires that the nodes have synchronized clocks[11]. However, that technique can only discover attacks long after they happened.

Hauser et al[12] avoid that defect by using hash chains to reveal the status of specific links in a link-state algorithm. Their method also requires synchronization of the nodes.

Hu[13] introduced another technique called SEAD that uses a node-unique hash chain that is divided into segments. The segments are used to authenticate hop counts. However, DSDV distributes routing information only periodically.

In many applications, reactive or on demand routing protocols are preferred. With on demand routing, source nodes request routes only as needed. On demand routing protocols performs better with significantly lower overhead than periodic routing protocols in many situations [13]. The authentication mechanism of Ariadne[13] is based on TESLA[15]. They use only efficient symmetric-key cryptographic primitives. The main drawback of that approach is the requirement of clock synchronization, which is very hard for wireless ad hoc networks.

Most secure routing protocols are based on authentication in the route discovery process. Some techniques detect faulty links based on observation of misbehavior during packet forwarding.

Marti et al [16] described a protocol for detecting and avoiding routers that drop or modify packets in ad hoc networks running DSR protocol. They have trusted nodes monitoring neighboring nodes. That technique does not work well in multi-rate wireless networks because nodes might be able to intercept packets forwarded with different modulations schemes. In addition, that method is vulnerable to collusion and misbehavior because there is no authentication.

Awerbuch et al[17] invention was based on adaptive probing techniques. However, malicious nodes can differentiate probing packets from normal data packets, and therefore, can selectively forward the probing packets to avoid detection.

Herzberg et al[18] described a combination of acknowledgements, timeouts and fault announcements, to detect packet forwarding faults. This proposal empirically described by Avramopoulos et al[19]. However, that protocol requires a separate authentication password for each of the intermediate router, thus adding more communication overhead when multi-hops are used.

A secure dynamic routing (SDR)[20] protocol is entirely on demand, and uses two primary mechanisms, route discovery and route maintenance. When a source node has a packet to send to a destination node but does not have a route to that destination node, the source node broadcasts a route request (RREQ) packet. The packet specifies the destination and a unique RREQ broadcast identifier. A receiving node attaches its own node address to a list in the RREQ and rebroadcast the RREQ. When the RREQ reaches the destination node, or any intermediate node that knows a route to the destination, that node sends a route reply (RREP) packet back to the source node, including an accumulated list of addresses from the source to the destination node. When the RREP reaches the source node, it stores the route in its route cache. Route maintenance is a mechanism for detecting changes in the topology of the network that can make a stored route invalid. This is done with a route error packet.

IV. SEAD-FHC

a) An algorithmic description of the SEAD-FHC

1. A method authenticates packets that are transmitted serially in a network.
2. A current password is selected for a current packet to be transmitted.
3. p_c Includes current data d_c.
4. A secure hash function $f_{(h)}$ is applied to the pw_c current password to form a current tag t_c.
5. A password pw_n is selected for a packet p_n that is in sequence and fallows p_c, which includes data d_n, and $f_{(h)}$ is applied to pw_n to form a tag t_n.
6. $f_{(h)}$ Is then applied to the d_n, t_n and pw_c to obtain a hashed value H_c.
7. p_c Is then transmitted that includes the H_c, d_c, t_c, password pw_p of packet p_p that sent before p_c in sequence to authenticate d_c.

b) Algorithm to authenticate sequence transmission of the packets

Countersign $cs_{(p_c)}$ will be selected for p_c that includes d_c to be transmitted.

$$t_c = f_{(h)}(cs_{(p_c)})$$

Countersign $cs_{(p_n)}$ will be selected for p_n with data d_n to be transmitted in sequence,

$$t_n = f_{(h)}(cs_{(p_n)})$$

Apply $f_{(h)}$ to the d_n, t_n and $cs_{(p_c)}$ that creates authentication tag for p_n referred as at $_{(p_n)}$

$$at_{(p_n)} = f_{(h)}(< d_n, t_n, cs_{(p_c)} >)$$

Transmit p_c from a source node n_s to a destination node n_d through hops in path selected through optimal route selection strategy.

The currently transmitting packet contains $at_{(p_n)}, d_c, t_c$ and a countersign $cs_{(p_p)}$ of packet p_p that transmitted before p_c to authenticate d_c.

In the interest of route maintenance, every hop in rout contains a cache that maintains hop list describing the route selected using an optimal route selection model. We apply $f_{(h)}$ on cache of each hop of the route to verify the integrity of the hop list cached.

c) Architecture of the proposed protocol

Proposed model provides an authentication protocol for a wireless ad hoc network where packets are transmitted serially. By serially, we mean a current packet p_c is immediately preceded by a previous packet p_p, and followed immediately by a next packet p_n.

More particularly, during a route discovery phase, we provide secure route selection, i.e., a shortest intact route, that is, a route without any faulty links. During route maintenance phase, while packets are forwarded, we also detect faulty links based on a time out condition. Receiving an acknowledgement control packet signals successful delivery of a packet.

For packet authentication, we use $f_{(h)}$ described by Benjamin Arazi et al [21]. The hash function encodes a countersign to form a tag.

By $f_{(h)}$ we mean that the countersign cannot be decoded from the tag and the countersign is used only once, because part of its value lies in its publication after its use. We have adapted that protocol for use in an ad hoc network where multiple packets need to be sent sequentially. Therefore, if a number of packets are sent sequentially, the countersign needs to be refreshed each time. Thus, a single authentication is associated with a stream of future packets that is significant difference between proposed and existing hash chain techniques. The existing models require stream of future events. In addition, the countersign is used to authenticate p_c but not for future packets.

Global Journal of Computer Science and Technology Volume XI Issue XX Version I December 2011

As an advantage over prior art asymmetric digital signature or secret countersigns do not need to be known ahead of time or distributed among the nodes after the system becomes operational. It should also be noted, that each countersign is used only one time, because the countersign is published to perform the authentication.

The $f_{(h)}$ as implemented by the proposal is ideal for serially communicating packets along a route in an ad hoc network, without requiring the nodes to establish shared secret countersigns beforehand.

The protocol includes the following steps. Select a random countersign cs_r. Form a tag t_r, $t_r = f_{(h)}(cs_r)$, Construct a message mac_r Form a hash value $H_r = f_{(h)}(< mac_r, t_r, cs_r >)$, and make it public. Perform the act and reveal mac_r, t_r, cs_r to authenticate the act.

V. Simulation and Results Discussion

The experiments were conducted using NS 2. We build a simulation network with hops under mobility and count of 100 to 1400. The simulation parameters described in table 1. We assume that each node has a memory buffer large enough to ensure that normal packets are never dropped because of congestion. Authentication ensures that the buffer is properly allocated to valid packets. Buffers also protect against traditional DoS, in which malicious nodes flood the network with unauthenticated packets. Malicious nodes that send packets frequently could otherwise quickly consume all allocated buffer space.

We authenticate route request (RREQ) by $f_{(h)}$ at source node and broadcast identifier in the route discovery phase, and data control packets in the packet forwarding phase. Thus, we prevent malicious requests and replay attacks. We also use a per-hop hashing to verify that no intermediate hop is omitted in a node list describing a route. A route reply (RREP) is authenticated by a destination node and therefore, attackers cannot cheat other nodes by fabricating routing information.

Table1 : Simulation parameters that we considered for experiments

Number of nodes	100 to 1400
Maximum velocity	20 m/s
Dimensions of space	1500 X 300 m2
Nominal radio range	250 m
Source destination pairs	20
Source data rate (each)	4 packets/s
Application data payload size	512 bytes/packet
Total application data load	327 kbps
Raw physical link bandwidth	2 mbs
Periodic route update interval	15s
Periodic updates missed before link is declared broken	3
Maximum packets buffered per node per destination	5
Hash length	80 bits

Our authentication mechanism is different than existing secure routing protocols based on digital signature, because only efficient symmetric key cryptography is used. Our method is also better than existing hash chain based protocols, because a node stores only one countersign, while hash chain based protocols store multiple countersigns, which increases memory requirements.

To detect faulty links, we use acknowledgements, timeouts, and fault announcements; these can also be authenticated by our $f_{(h)}$. Therefore, we need only a single authentication tag for each data and control packet; thereby bandwidth and memory usage is low.

With faulty link detection, all passive and active attackers that fail to forward data packets and that maliciously misdirect data packets are recognized and avoided in subsequent routings.

a) Protocol Description

i. *Secure Route Discovery*

In on demand routing protocols, e.g., DSR, a source node initiates route discovery to find a route when the source node has a packet to send to a destination node, and the source node does not store a route to the destination node in its route cache. The source node does this by broadcasting a RREQ control packet to neighboring nodes. Neighboring nodes rebroadcast the request, until the request eventually finds its way to the destination node so that intermediate nodes on the route can be discovered. We authenticate the RREQ control packet with hash function $f_{(h)}$.

ii. *RREQ Authentication*

The routing path between source node n_s and destination node n_d contains $n_h, n_{h+1} .. n_{h+z}$ as intermediate hops, where 'z' is count of the intermediate hops.

The source node selects two random countersigns cs_r and $cs_{r'}$, and broadcasts a first RREQ:

$$
\begin{aligned}
& n_{s(id)} =< n_s(a_{id}), n_s(b_{id}) = 1 > \\
& sig_{n_s} = f_{(ds)}(n_s(id), f_{(h)}(cs_r) \\
& n_{s+1(id)} =< n_{s+1}(a_{id}), n_{s+1}(b_{id}) = 2 > \\
& RREQ_i = \{n_{s(id)}, f_{(h)}(cs_r), sig_{n_s}), f_{(h)}(n_{s+1(id)}, f_{(h)}(cs_{r+1}), cs_r), n_d(a_{id}), f_{(h)}(n_s, n_d)\}
\end{aligned}
\tag{1}
$$

In Eq(1) $f_{(ds)}$ is optimal digital signature function, a_{id} is node address identity and b_{id} is broadcast id.

$f_{(ds)}(n_s(id), f_{(h)}(cs_r))$ is a digital signature to verify $(n_s(id), f_{(h)}(cs_r)$ by other nodes, so that every intermediate node and the destination can verify that the $(a_{id}, b_{id}, f_{(h)}(cs_r))$ in the $RREQ_i$ packet is valid and indeed generated by the claimed n_s.

A hop node in the route path generates a route entry by storing the a_{id} of n_s, $b_{id} = 1$, $hcs_r = f_{(h)}(cs_r)$, and $h_{e2} = f_{(h)}(n_{s+1}, f_{(h)}(cs_{r'}), cs_r)$. These values can verify future route requests from the same source node. The component $< a_{id}, b_{id} >$ uniquely identifies $RREQ$.

The value b_{id} is incremented whenever the source node issues a new $RREQ$.

The secret key $k_{(n_s, n_d)}$ is shared between n_s, n_d. This needs only be used for the first packet.

Because of the broadcast nature of the $RREQ$ control packets, every node in the ad hoc network eventually receives the $RREQ_i$ after a time '$m\Delta$', where m is a diameter of the network, and Δ is a maximum delay at intermediate hop.

After a time interval '$m\Delta$', the n_s sends next route request $RREQ_{i+1}$. Therefore, the source node selects next random countersign cs_{r+2}, and broadcasts $RREQ_{i+1}$:

$$
\begin{aligned}
& n_{s+1(id)} =< n_{+1}(a\ \ ), n_{+1}(b\ \ ) = 2 > \\
& cs_r \ \ = f_{(ds)}(n_{s+1}(id), f_{(h)}(cs_{r+1})) \\
& n_{s+2(id)} =< n_{s+2}(a_{id}), n_{s+2}(b_{id}) = 3 > \\
& RREQ_{i+1} = \{n_{s+1(id)}, f_{(h)}(cs_{r+1}), sig_{n_{s+1}}), f_{(h)}(n_{s+2(id)}, f_{(h)}(cs_{r+2}), cs_{r+1}), n_d(a_{id}), f_{(h)}(n_s, n_d)\}
\end{aligned}
\tag{2}
$$

The intermediate node finds the route entry associated with the claimed source node, and performs $f_{(h)}$ on cs_r that received in $RREQ_{i+1}$ and checks the equality with hcs_r that received in $RREQ_i$, which stored in the route entry. If $f_{(h)}(cs_r)$ is equal to hcs_r then n_h applies $f_{(h)}$ on $(n_{s+1(id),} f_{(h)}(cs_{r+1}), cs_r)$ that received through $RREQ_{i+1}$ and checks if the result is the same as h_{e2} stored in the route entry, if valid, the authenticity of $(n_{s+1}, f_{(h)}(cs_{r+1}))$ is verified. Thus, n_h is assured that $RREQ_{i+1}$ is from the claimed source node and the present b_{id} is valid. The n_h then updates its routing entry by recording b_{id} that received through $RREQ_{i+1}$, $hcs_r = f_{(h)}(cs_{r+1})$ and $h_{e2} = (n_{s+2}, f_{(h)}(cs_{r+2}), cs_{r+1})$, which are used to authenticate $RREQ_{i+2}$.

In general, before sending a k^{th} route request $RREQ_k$, the source node waits a time interval $m\Delta$ after sending the previous request $RREQ_{k-1}$. Then, the source node selects a new random countersign cs_{k+1}, and broadcasts $RREQ_k$:

As a part of process at n_h, appends its own address to the intermediate node list in the $RREQ$, performs the per-hop hashing, which is achieved by calculating a new hash tag by hashing its own address concatenated with the old hash tag, and replacing the old hash tag, then rebroadcasts the RREQ. If any check fails, the RREQ is dropped.

Thus, with per-hop hashing, an attacker cannot delete an intermediate node from the node list, because the attacker does not have the secret countersign between the intermediate node and the destination node.

When the $RREQ$ reaches the 'n_d', then n_d verifies it by checking if $k_{(n_h,n_d)}$ If the check succeeds, then the integrity of this $RREQ$ is verified, along with the authenticity of its origin and every intermediate node along the path from node n_s to node n_d.Then n_d sends a $RREP$ back to the source node, including an authenticated copy of the accumulated list of addresses from the $RREQ$ i.e., the packet data for the $RREQ$ control packet.

The $RREP$ control packet contains

$$n(lst)_h = < n_h, n_{h+1}, .. n_{h+z} > ... where\ 'z' represents number\ of\ intermediate\ hops$$

$$< b_{id}, (n_s, n(lst)_h, n_d), f_{(h)}(n_d, n_{h+z}), f_{(h)}(n_d, n_{h+z-1}), f_{(h)}(n_d, n_{h+z-2}), ..., f_{(h)}(n_d, n_{h+1}), f_{(h)}(n_d, n_h), f_{(h)}(n_d, n_s) >$$

Where b_{id} is for the source n_s to verify the freshness of the reply. .As the $RREP$ packet passes through intermediate nodes back to the source node, each node checks the corresponding authentication tag, and stores the route information in its route cache. The source node then selects a shortest route to the destination node without previously detected faulty links.

iii. *Data transmission and malicious hop detection*

Here in this section we describe the procedure of authentication data packets forwarded from the source node to the destination node, along the selected route, while checking for faulty links. In DSR, the source route information is carried in each packet header.

To send a packet m_i that is a part of data to be sent to destination node n_d, the source node n_s picks two counter signs cs_r, cs_{r+1} and fixes the time limit to receive either one of packet delivery acknowledgement ack or a control packet mn_{ack} that acknowledges about malicious link in the route path. The source node sends message with the format

$$msg_i = \{m_i, f_{(h)}(cs_r), f_{(ds)}(m_i, f_{(h)}(cs_r)), f_{(h)}(m_{i+1}, f_{(h)}(cs_{r+1}), cs_r)\}$$

to the n_h along the route.

Here $f_{(ds)}(m_i, f_{(h)}(cs_r))$ is a digital signature to verify $(m_i, f_{(h)}(cs_r))$ by intermediate hops of the route selected, so that every 'n_h' and 'n_d' can verify that $(m_i, f_{(h)}(cs_r))$ is valid and indeed generated by the claimed n_s.

Then each hop updates route table entry for source node S by recording $f_{(h)}(cs_r)$ as $hcs_r(n_s)$, $f_{(h)}(m_{i+1}, f_{(h)}(cs_{r+1}), cs_r)$ as $h_{e2}(n_s)$, which is used to authenticate an immediate fallowing message msg_{i+1} in sequence.

When sending the data packet m_{i+1}, the n_s selects another countersign cs_{r+2} and forwards the msg_{i+1} to the first hop of the selected path:

$$msg_{i+1} = \{m_{i+1}, f_{(h)}(cs_{r+1}), cs_r, f_{(h)}(m_{i+2}, f_{(h)}(cs_{r+2}), cs_{r+1})\}$$

Each node on the route calculates $f_{(h)}(cs_r)$ and compares with $hcs_r(n_s)$ that available in routing table, if results equal then cs_r will be authenticated as valid. The n_h then calculates $f_{(h)}(m_{i+1}, f_{(h)}(cs_{r+1}), cs_r)$, and compares with $h_{e2}(s_n)$ result is equivalent then claims the validity of $(m_{i+1}, f_{(h)}(cs_{r+1}))$. The node then updates its routing entry by recording $hrc_{r+1} = f_{(h)}(rc_{r+1})$ and $h_{(e2)}(n_s) = f_{(h)}(m_{(i+2)}, f_{(h)}(cs_{r+2}), r_{r+1})$, and and forwards the data packet to the node along the route as specified in the header of the packet header.

During the packet sending process described earlier, if any of the checks fails, then the packet is dropped. If both checks succeed, then the node updates its routing entry associated with n_s. If the check at n_h, then either n_{h-1} or $f_{(h)}(m_{i+1}, f_{(h)}(cs_{r+1}), cs_r)$ in msg_i has been modified, or node n_{h-1} modified $f_{(h)}(m_{i+1}, f_{(h)}(cs_{r+1}), cs_r)$ in msg_{i+1}. In either case, the current hop node n_h drops the packet. Consequently, hop node n_{h-1} does not receive a valid ack after time out, and the node can report a malicious activity at (n_{h-1}, n_h) connection, or the hop node n_{h-2} reports about malicious activity between (n_{h-2}, n_{h-1}) to n_s. In either case, the fault link includes the malicious node n_{h-1}.

In our proposed model the authentication tag of each packet limited to two hashes and one countersign; while in the existing models required N authentication tags for a route with N hops. Therefore, our method has a lower communication and storage overhead.

The packet authentication process at n_d is identical to the authentication process at any intermediate hop n_h. If any of the checks fails, then the packet is dropped. If both checks succeed, the packet is delivered successfully, and schedules the 'ack' for transmission along the reverse of path of the route. The ack reflects the packet identification number i.

The destination node also appends an authentication tag to the ack message for the nodes on the reverse path. The authentication tag bears the same structure as the one generated by the source node. Specifically, when sending ack_i, for the packet 'm_i', the destination node randomly selects two countersigns cs_{re} and cs_{re+1}, and sends the following information:

$$ack_i,\ f_{(h)}(cs_{re}),\ f_{(ds)}(ack_i,\ f_{(h)}(ack_i)),\ f_{(h)}(ack_{i+1},\ f_{(h)}(cs_{re+1}), cs_{re}).$$

Similarly, $f_{(ds)}(ack_i, f_{(h)}(cs_{re}))$ is used to verify $(ack_i, f_{(h)}(cs_{re}))$ by each node along the reverse path of the route. When sending the acknowledgement for packet 'm_i', the destination selects a new countersign cs_{re+1} and forwards:

$$(ack_{i+1},\ f_{(h)}(cs_{re+1}), cs_{re},\ f_{(h)}(ack_{i+2}\ f_{(h)}(cs_{re+2}), cs_{re+1})).$$

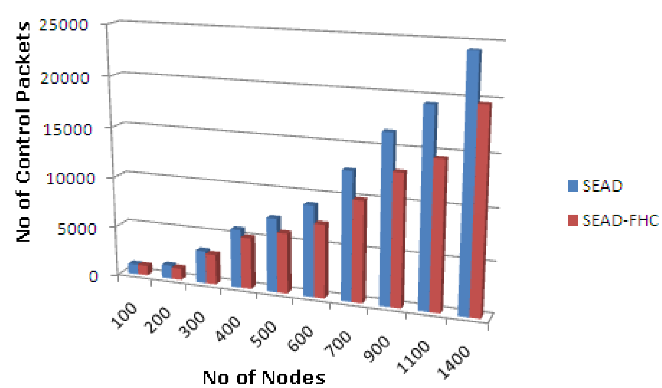
If the timeout at an intermediate node expires, then that node sends mn_{ack} with an identification number according to our hash function for authentication of the mn_{ack} by the upstream nodes. When a node receives the ack, the node verifies its authenticity and that a timeout is pending for the corresponding data packet. If the 'ack' is not authentic or a timeout is not pending, the node discards the ack. Otherwise; the node cancels the timeout and forwards the 'ack to the next node.

When a node receives mn_{ack}, it verifies its authenticity, and that a timeout is pending for the corresponding data packet, and that the link reported in the mn_{ack} is the first downstream to the node that generated mn_{ack}. If the mn_{ack} is not authentic, or a timeout is not pending, or the link is not the downstream to the node reporting 'mn_{ack}', then the node drops mn_{ack}. Otherwise, the node cancels the timeout and further forwards the mn_{ack} control packet. Upon receiving 'mn_{ack}', the source node deletes the link that connecting n_h referred in mn_{ack} and finds a new route. In this proposed model, the packets are always received as in the order they sent. This is because all packets are forwarded along the same route in DSR. In the case of congestion and buffering, the messages are stored in a first-in-first-out buffer according to the order that they are received.

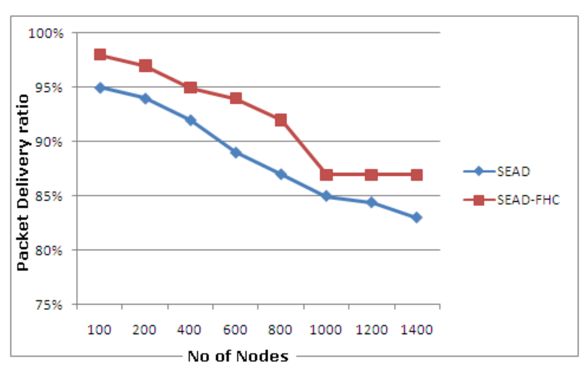
When the source node wants to use another path to the destination node, the source node selects a new countersign and authenticates the countersign with every node along the new route, and reinitiates the entire process, as described above.

b) Results Discussion

Here we describe the scalability of SEAD-FHC over SEAD in terms of control packet that costs resource utilization. We can observe that SEAD-FHC is almost similar to SEAD when node count is fewer. But we can observe that SEAD-FHC improving the minimization of the control packets when node count increased. It is obvious since the SEAD-FHC stabilizing the delay time even at maximum node count, which helps in minimizing packet drops due to delay and improves throughput. This results as fewer control packet utilization.

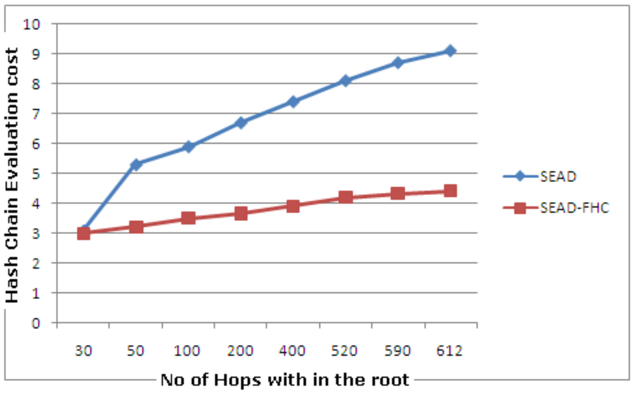


Here we describe the scalability of SEAD-FHC over SEAD in terms of packet delivery ratio. Since Hash chain computation cost is drastically minimized in SEAD-FHC, the delay time minimized and throughput increased.



Here we describe the performance of SEAD-FHC over SEAD in terms of Hash chain evaluation cost. Let λ be the cost threshold to evaluate each hash in hash chain. We measure the Hash chain evaluation cost as

$\frac{\sum_{i=1}^{z}\sum_{j=1}^{n}\lambda}{z}$, here z is number of nodes and n is number of hashes, as of the chaining concept of SEAD z=n but in SEAD-FHC n always 2



VI. Conclusion

Here in this paper we proposed a secure efficient distance vector routing with fixed hash chain length in short we referred as SEAD-FHC. We argued that fixed hash chain limits the computation cost and resource utilization. We empirically demonstrated that SEAD-FHC is scalable and performs well over SEAD. In future experiments can target to extend this protocol to support path restoration mechanism. Here SEAD-FHC relies on new route detection upon link failure.

References References Referencias

1. Perkins: "Ad hoc On-Demand Distance Vector Routing," Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, February 1999.
2. Wood: "Denial of Service in Sensor Networks," IEEE Computer Magazine, October 2002.
3. Perlman: "Network Layer Protocols with Byzantine Robustness," Ph.D. thesis, MIT LCS TR-429, October 1998.
4. Papadimitrators: "Secure Routing for Mobile Ad Hoc Networks," SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, January 2002.
5. "The Keyed-Hash Message Authentication Code (HMAC)," No. FIPS 198, National Institute for Standards and Technology (NIST), 2002.
6. Dahill: "A Secure Routing Protocol for Ad Hoc Networks," Technical Report UM-CS-2001-037, University of Massachusetts, August, 2001.
7. Zapata: "Secure Ad hoc On-Demand Distance Vector Routing," ACM Mobile Computing and Communications Review (MC2R), July 2002.
8. Rivest: "A method for obtaining Digital Signatures and Public Key Cryptosystems," Comm. of ACM, February 1978,
9. Lamport: "Password Authentication with Insecure Communication," Comm. of ACM, November 1981.
10. Lamport: "Constructing Digital Signature Based on a Conventional Encryption Function", 1979.
11. Cheung: "An Efficient Message Authentication Scheme for Link State Routing", Computer Security Applications Conference, 1997.
12. Hauser: "Reducing the Cost of Security in Link State Routing," Symposium on Network and Distributed Systems Security, February 1997.
13. Hu: "Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks", MobiCom, September 2002.
14. Y. C. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," Ad Hoc Networks Journal, 1, 2003, pp.175-192.
15. Perrig: "Efficient and Secure Source Authentication for Multicast," Network and Distributed System Security Symposium, February 2001.
16. Marti: "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," ACM International Conference on Mobile Computing and Networking, August 2000.
17. Awerbuch: "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures," ACM Workshop on Wireless Security, September 2002.
18. Herzberg : "Early Detection of Message Forwarding Faults," SIAM J. Comput., Vol. 30, no. 4, pp. 1169-1196, 2000.
19. Avramopoulos: "A Routing Protocol with Byzantine Robustness," The 2003 IEEE Sarnoff Symposium, March 2003.
20. Johnson: "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, Kluwer Academic Publishers, 1996.
21. Benjamin Arazi: "Message Authentication in Computationally Constrained Environments", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 8, NO. 7, JULY 2009.
22. B. Bellur, R.G. Ogier, A reliable, efficient topology broadcast protocol for dynamic networks, in: Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 99), March 1999, pp. 178–186.
23. R.V. Boppana, S. Konduru, An adaptive distance vector routing algorithm for mobile, ad hoc networks, in: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), 2001, pp. 1753–1762.
24. T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized Link

State Routing Protocol, Internet-draft, draft-ietf-manet-olsr- 05.txt, October 2001, Work in Progress.

25. J.J. Garcia-Luna-Aceves, C.L. Fullmer, E. Madruga, D. Beyer, T. Frivold, Wireless Internet Gateways (WINGS), in: Proceedings of IEEE MILCOM 97, November 1997, pp. 1271–1276.
26. J. Jubin, J.D. Tornow, The DARPA Packet Radio network protocols, Proceedings of the IEEE 75 (1) (1987) 21–32.
27. S. Murthy, J.J. Garcia-Luna-Aceves, An efficient routing protocol for wireless networks, Mobile Networks and Applications 1 (2) (1996) 183–197.
28. C.E. Perkins, P. Bhagwat, Highly Dynamic Destination- Sequenced Distance-Vector routing (DSDV) for mobile computers, in: Proceedings of the SIGCOMM 94 Conference on Communications Architectures, Protocols and Applications, August 1994, pp. 234–244.
29. D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Dordrecht, 1996, pp. 153–181.
30. V.D. Park, M.S. Corson, A highly adaptive distributed routing algorithm for mobile wireless networks, in: Proceedings of INFOCOM 97, April 1997, pp. 1405–1413.
31. Z.J. Haas, A routing protocol for the reconfigurable wireless network, in: 1997 IEEE 6th International Conference on Universal Personal Communications Record: Bridging the Way to the 21st Century (ICUPC 97), vol. 2, October 1997, pp. 562–566.
32. C. Hedrick, Routing Information Protocol, RFC 1058, November 1988.
33. G.S. Malkin, RIP version 2 protocol applicability statement, RFC 1722, November 1994.
34. G.S. Malkin, RIP version 2, RFC 2453, November 1998.
35. N. Abramson, The ALOHA system—another alternative for computer communications, in: Proceedings of the Fall 1970 AFIPS Computer Conference, November 1970, pp. 281–285.
36. F. Baker, R. Atkinson, RIP-2 MD5 Authentication, RFC 2082, January 1997.
37. S. Basagni, K. Herrin, E. Rosti, D. Bruschi, Secure Pebblenets, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), Long Beach, CA, October 2001, pp. 156–163.
38. J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J.G. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98), October 1998, pp. 85–97.

Global Journal of Computer Science and Technology Volume XI Issue XX Version I December 2011