SecMR – a secure multipath routing protocol for ad hoc networks

Rosa Mavropodi *, Panayiotis Kotzanikolaou, Christos Doulgeris

Department of Informatics, University of Piraeus, 80 Karaoli & Dimitriou, 18534 Piraeus, Greece

Available online 28 June 2006

Abstract

Multipath routing in ad hoc networks increases the resiliency against security attacks of collaborating malicious nodes, by maximizing the number of nodes that an adversary must compromise in order to take control of the communication. In this paper, we identify several attacks that render multipath routing protocols vulnerable to collaborating malicious nodes. We propose an on-demand multipath routing protocol, the secure multipath routing protocol (SecMR), and we analyze its security properties. Finally, through simulations, we evaluate the performance of the SecMR protocol in comparison with existing secure multipath routing protocols.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Multipath routing protocol; Security; Collaborative attacks

1. Introduction

In the mobile ad hoc network paradigm routing is a challenging task due to mobility and the resulting inherent dynamic network topology. Moreover, the nodes in ad hoc networks are usually restricted devices in respect to their energy sources, computational capabilities and communication range. Ad hoc routing protocols may be generally categorized as table driven (often called proactive) and source initiated (or on-demand). In table driven protocols (e.g. ZRP [7]), each host continuously maintains complete network routing information. On-demand schemes (e.g. DSR [10]) invoke the routing discovery process only on demand, in a query/reply approach.

According to the number of paths that are discovered from a route request, the routing protocols are divided into single path (e.g. [10,21]) and multipath (e.g. [22,18]). Another feature of the routing protocols is the number of the discovered paths that are actually used for sending data. Some protocols use only a single path for the communication, while others distribute the data through different channels. The route discovery process in the multipath protocols may be initiated either when the active path collapses (in that case communication is performed with one of the alternative paths), or when all known paths towards the destination are broken [13]. The route discovery may stop when a sufficient number of paths are discovered or when...
all possible paths are detected. The protocols of the second case are also known as complete. Multipath routing protocols can be node-disjoint [23] or link-disjoint [14] if a node (or a link) cannot participate in more than one path between two end nodes.

Routing security is a challenging task in mobile ad hoc networks since the lack of fixed infrastructure makes routing an obvious target for adversaries. Several solutions for secure routing have been proposed, such as collaborative monitoring of the routing behavior between nodes [16,24], motivating nodes to behave well with fictitious currency [4] or participation of nodes in routing paths based on quantifiable criteria [25]. However, these solutions cannot resist Denial-of-Service (DoS) attacks of malicious nodes, since the protocols are designed for single path routing protocols.

Multipath routing protocols are resilient to DoS attacks and may protect network availability from faulty or malicious nodes [2]. By employing $k$ node-disjoint routing paths between two communicating nodes the integrity and availability of the communication is maximized, since the routing protocol is resilient against DoS attacks of an adversary that controls less than $k$ malicious nodes.

The SRP [19] is a multipath routing protocol which aims at this kind of protection. SRP uses end-to-end symmetric cryptography, in order to protect the integrity of the route discovery. Thus, it is very efficient and it protects from several attacks of malicious nodes. However, the route request propagation is inherently weak to the racing phenomenon, which may prevent the discovery of existing node-disjoint paths. Moreover, the intermediate nodes are not authenticated, making the protocol vulnerable to impersonation and sybil attacks [5]. Thus, a malicious node may participate with fake identities to several paths, rendering multipath routing insecure.

In this paper, we propose a complete on-demand multipath routing protocol, the Secure Multipath Routing (SecMR) protocol, which provides protection against DoS attacks from a bounded number of collaborating malicious nodes. Preliminary work on the SecMR protocol and its properties have been presented in [12,17]. In Section 2, we indicate vulnerabilities of multipath protocols which make them more vulnerable than expected, to collaborating malicious nodes. In Section 3 we describe the SecMR protocol in detail. Section 4 analyzes the security characteristics of our protocol and Section 5 discusses the efficiency of SecMR in comparison with other secure multipath routing protocols. Section 6 discusses on the combination of security and efficiency of the existing secure multipath routing protocols and concludes the paper.

2. Vulnerabilities of multipath routing

Several vulnerabilities may affect the route discovery of multipath routing protocols, allowing a small set, or even a single malicious node, to control the routing paths of selected nodes. These vulnerabilities include the racing phenomenon [12,8], lack of authentication [5] and vulnerabilities against Man-in-the-Middle attacks [1] caused by invisible nodes [15]. The security of multipath protocols that are subject to these vulnerabilities is thus reduced to the security of single path routing protocols.

3. Description of the SecMR protocol

SecMR discovers the complete set of the existing non-cyclic, node-disjoint paths between a source and a target node, for a given maximum hop distance. The protocol works in two phases. The first phase of the SecMR, the neighborhood authentication phase, involves the asynchronous mutual authentication of neighboring nodes. The second phase of the SecMR, the route discovery and maintenance phase, involves the establishment and maintenance of active routes.

3.1. Neighborhood authentication phase

Let $n_i$ denote a mobile node of an ad hoc network. We assume that each node $n_i$ possesses a pair of public-secret keys, $(PK_i, SK_i)$ respectively, of an Elliptic Curve Cryptosystem [11]. Furthermore, the public key of each node $n_i$ is certified through a certificate $cert_i$, issued by a Certifying Authority CA.
Although the use of Certificate Authorities and PKI in a mobile ad hoc environment is a challenging task with still many open issues, several approaches have been proposed, e.g. [27,9,26]. The main idea behind these approaches is the implementation of a distributed PKI by using $k$-out-of-$n$ threshold cryptography and/or localized distributed trust between peer nodes. The certification and revocation of the public keys is out of the scope of this paper. In our work we assume that the nodes are able to use the services of a (distributed) CA, which may be implemented by any appropriate technique.

The certificate $cert_i$ also contains a unique identifier $ID_i$, which is assigned by the CA to each node $n_i$. According to the size of the network, the node identifier may be a relatively small number, e.g. 1 or 2 bytes. Of course, for large scale ad hoc networks a 2-byte node identifier may be insufficient. However, the use of a unique identifier scales well even for very large scale networks, since with a 2-byte identifier we can have up to 65,535 nodes, while with a 3-byte identifier we can have more than 16 million nodes. Keeping identifiers as small-sized as possible, helps in maintaining small-sized lists inside the request queries.

In periodic time intervals, each node $n_i$ broadcasts to its one-hop neighbors a signed message including the current time and its unique identifier that is included in its certificate, i.e.: $n_i \Rightarrow (t, ID_i, sig_i(t, ID_i), cert_i)$, where $X \Rightarrow m$ denotes that a node $X$ broadcasts a message $m$ at time $t$. Thus, each node will generate one signature and will verify one signature for each one of its neighbors. The average cost for this phase is one signature generation per node and $\mathcal{C}$ signature verifications per node, where $\mathcal{C}$ is the average network connectivity. The duration of the time period of the neighborhood authentication phase is a system parameter and depends on the volatility of the environment. According to the frequency of changes in the connectivity, the time period should be as long as possible.

After the verification phase, each node $n_i$ will generate a list of its neighborhood for the current time $t$, denoted as $N_i^t$, where a node identifier $ID_j \in N_i^t, 1 \leq j \neq i \leq L$ if at time period $t$, the node $n_i$ received an authentication message from node $n_j$, containing a valid signature of $n_i$ on $t, ID_j$. For simplicity we drop the time index, unless this is required for clarification. Thus, $N_i^t = N_i$. Note that the nodes do not re-broadcast the received authentication messages. Since authentication is performed in local neighborhoods and it is not executed simultaneously from all the nodes, the cost is tolerable and it does not lead to the storm problem.

3.2. Route discovery and maintenance phase

The route discovery and maintenance phase consists of three algorithms. The route request query algorithm is used to discover the node-disjoint paths between a source node $S$ and a target node $T$. The route reply algorithm is used to forward the routing paths back to the source node. Finally, the route error algorithm is used in order to update a broken or flawed path with a new one.

3.2.1. Route request query algorithm

When a source node $S$ wants to communicate with a target node $T$, it first checks whether it already has active routes in its routing table towards $T$. If not, it generates a route request query $Q_{S,T}$ as follows:

$$Q_{S,T} = [ID_S, ID_T, seq, hop_{cnt}, hop_{max}, E_{PK_T}(K_{S,T}), RouteList, ExcludeList, NextHop, hash_{K_{S,T}}(ID_S, ID_T, seq, hop_{max})],$$

where $ID_S$, $ID_T$, are the identifiers of $S$ and $T$ respectively; $seq$, is a counter used by $S$ for each new query; $hop_{cnt}$, is a counter that tracks the current number of hops; $hop_{max}$, is the maximum allowed hop distance; $E_{PK_T}(K_{S,T})$, is the encryption of the key $K_{S,T}$ with the public key $PK_T$ of node $T$; RouteList, is a dynamically generated list of the intermediate nodes participating in a path between $S$ and $T$; ExcludeList, is a dynamically generated list of nodes that are excluded for a particular thread of the query; NextHop, is the list containing the nodes that are allowed to be the next hop of the particular query, and $hash_{K_{S,T}}(ID_S, ID_T, seq, hop_{max})$, is the result of a keyed hash function with the key $K_{S,T}$.

Route request query algorithm on the source node $S$

The source node $S$ executes the algorithm shown in Fig. 1. The source node $S$ sets $hop_{cnt} = 0$ and selects the maximum hop number $hop_{max}$, based on the current knowledge of the network such as its connectivity, the expected node density etc. Then, it initializes the RouteList and the ExcludeList as null and it sets the initial list of the allowed next hops as $NextHop = N_S$, where $N_S$ is the most recent
neighborhood of $S$ that has been computed in the previous phase. Finally, $S$ selects a random key $K_{S,T}$, computes the encryption of this key with the public key $PK_T$ of the target node and constructs and broadcasts the route request query $Q_{S,T}$ as described above. For integrity check, the query also contains a keyed hash-value generated with the security association $K_{S,T}$, over the static fields of $Q_{S,T}$.

**Route request query algorithm on each intermediate node $n_i$**

An intermediate node $n_i$ receiving a thread of the request query $Q_{S,T}$ executes the algorithm shown in Fig. 2. Each node $n_i$ maintains a route table containing the current active routes towards other nodes. Furthermore, it maintains in the route table an index of the recently processed (and still not replied) queries, in order to avoid re-processing the same queries. To avoid maintenance of long route queries into the route table, each node that processes a query, it hashes the received query and stores the hashed value to its table for a reasonable time. Before processing a received request query $Q_{S,T}$, the node $n_i$ hashes $Q_{S,T}$ and checks whether the hash value belongs to its route table for a reasonable time. Before processing a received request query $Q_{S,T}$, the node $n_i$ hashes $Q_{S,T}$ and checks whether the hash value belongs to its route table (i.e., whether the specific instance of a route request has been already processed) in order to drop it. Note that this does not prevent the node $n_i$ from processing a different instance (thread) $Q'_{S,T}$ of the query containing the same values $ID_S$, $ID_T$ and $seq$, that reaches $n_i$ through a different path. Indeed, if the thread $Q'_{S,T}$ reaches $n_i$ from a different path, then the $RouteList$, $ExcludeList$ and $NextHop$ lists contained in $Q_{S,T}$ will be different from the corresponding lists contained in $Q'_{S,T}$. Thus, $\text{hash}(Q_{S,T}) \neq \text{hash}(Q'_{S,T})$ and each thread of the request coming from a different path will be processed once.

The algorithm prevents participation of a node identifier to more than one of the lists of a route request query. However, since this could happen due to faults or malicious acts, each node $n_i$ checks whether the lists of the received query have any common values, in order to drop such a query.

At this point, the node $n_i$ checks whether it is identified in the received $NextHop$ list of the request query, and also whether the sender of the request query, i.e., the last node identified in the received $RouteList$, belongs to its authenticated neighborhood $N_i$. If either of these checks fail, the request is dropped. This step provides dual authentication between any pair of intermediate nodes, as it is explained in the following section, which is critical for the protection against multiple identity attacks. After these checks, node $n_i$ is assured about the validity of the received request and is ready to process it. Node $n_i$ increases the hop counter $hop_{cnt}$ by one and checks the new value with the maximum allowed hop distance $hop_{max}$, in order to stop processing of long queries. This prevents the query from propagating in distant areas. Of course, this limitation will also prevent the discovery of any existing node-disjoint path that consist of more than $hop_{max}$ hops. Thus, this parameter should be selected with care, taking under consideration the current topology, the acceptable signaling delay etc.

The node $n_i$ processes the request query. The updated $RouteList$, is constructed by appending its

---

**Fig. 1. Route request query algorithm on the source node.**
1) If (hash(QS,T) ∈ RouteTable(n))
   /* Drop the request query if the particular query thread has already been received by n */
   OR ((RouteList \&\& ExcludeList ≠ ∅)
   OR (RouteList \&\& NextHop ≠ ∅)
   OR (ExcludeList \&\& NextHop ≠ ∅))
   /* Drop the query if the same node identifier belongs to more than one list */
   OR ((ID, ∈ NextHop) OR (LastElement(RouteList) ≠ N))
   /* Drop the query if the previous node is not an authenticated neighbor of n */
   then
   DROP(QS,T)
   else {
2) add(hash(QS,T), RouteTable(n))
   /* The node n marks the specific route request query as processed */
3) if (ID, = ID2) then REPLY(QS,T)
   /* If n is the last node, execute the route reply algorithm and exit */
   else {
4) hop_max = hop_max + 1
5) if (hop_max > hop_max)
   /* Drop the query if it exceeds the maximum allowed hop-distance, to prevent propagation
   of the request query to long distance areas */
   then
   DROP(QS,T)
   else {
6) RouteList = RouteList + ID,
   /* Node n adds itself to the RouteList */
7) ExcludeList = ExcludeList + (NextHop - ID)
   /* Node n excludes the rest of the neighbors of the previous node, from this particular
   thread of the route request query */
8) NextHop = N, - (N, \&\& RouteList) - (N, \&\& ExcludeList)
   /* The neighbors of n are the allowed next hops of this thread of the query, unless they
   already belong to the routing path or have already been excluded from the routing path */
9) Update and Broadcast
   Qs,T = [IDS, IDT, seq, hop_max, RouteList, ExcludeList,
   NextHop, EncPKk (Ks,T), hashKs,T (IDS, IDT, seq, hop_max)]
   /* Update the query with the new values and broadcast it */
   }
   }
}

Fig. 2. Route request query algorithm on intermediate nodes.

identifier IDj to the received RouteList. The updated ExcludeList is generated by appending the rest of
the nodes included in the received NextHop list, into the received ExcludeList (duplicates are removed).
Finally, the updated NextHop list is generated as the list of identifiers included in its own neighborhood
Ni (again, node identifiers already participating in another list are removed). Now, the node ni updates the query thread with the new values and broadcasts it.

The use of the ExcludeList is a key element for the efficient propagation of a route request query.
By dynamically generating threads of a request, the algorithm eventually discovers all the existing
node-disjoint paths of at most hop_max distance. To clarify the threading of a request query, consider
the following scenario. Let ni be the node that broadcasts the request Qs,T to its neighbors nj, nk
(Fig. 3). In order to distinguish the various threads of the request query, we denote as Qs,T/i the thread
that is processed by node ni. Thus, the thread Qs,T/i
broadcasted by node ni will contain the lists: RouteList = {X, IDi}, ExcludeList = {Y} and Next-
Hop = {IDj, IDk}, where X and Y denote sequences of node identifiers.

Both nodes nj and nk will process the request (supposing that IDj, IDk Ξ X, Y). Node nj will add
its identifier to the RouteList, add the identifier of
structs the maximum set of node-disjoint paths \(M\).

### Remark 1.
In order to avoid delays caused by the time required to compute the maximum set of node-disjoint paths, the target node \(T\) may use the first valid thread to construct a temporary path. When the set \(M\) of node-disjoint paths is computed, \(T\) may use all the corresponding paths.

#### 3.2.2. Route reply algorithm

When the target node \(T\) receives a thread of a route request query \(Q_{S,T/j}\), it decrypts \(E_{IK_s}(K_{S,T})\), obtains the key \(K_{S,T}\) and checks the validity of the included keyed hash-value. Then, it waits for a certain amount of time to receive any other threads of the same route request query coming from different paths. The keyed hash-value of each thread is also checked. Then, the target node \(T\) constructs the maximum set of node-disjoint paths \(M\) as follows:

Say that \(T\) received \(m\) valid threads \(Q_{S,T/j_1}, \ldots, Q_{S,T/j_m}\). From the corresponding lists \(RouteList_{j_1}, \ldots, RouteList_{j_m}\), node \(T\) computes the maximum set of lists with non-common values. Without loss of generality, we may say that the maximum set of lists with non-common values is \(\{RouteList_{j_1}, \ldots, RouteList_{j_k}: \bigcap_{i=1}^k RouteList_{j_i} = \emptyset\}\), \(1 \leq k \leq m\). Then, the maximum set of node-disjoint paths between \(S\) and \(T\) is \(M = \{p_1, \ldots, p_k\} = \{(ID_S, RouteList_{j_1}, ID_T), \ldots, (ID_S, RouteList_{j_k}, ID_T)\}\). For each \(RouteList_{j_i} \in M\), node \(T\) constructs and broadcasts a route reply message as: \(R_{S,T/j_i} = [ID_S, ID_T, seq, RouteList_{j_i}, hash_{K_{S,T}}(ID_S, ID_T, seq, RouteList_{j_i})]\).

Each intermediate node \(n_i\) that receives a route reply message \(R_{S,T/j_i}\) checks whether \(n_i \in R_{S,T/j_i}\). If not, then it drops the reply message. Otherwise, if \(ID_i \in RouteList_{j_i}\), then node \(n_i\) checks whether the nodes identified before and after \(ID_i\) in \(RouteList_{j_i}\) belong to \(N_i\). In that case, node \(n_i\) re-broadcasts the reply message \(R_{S,T/j_i}\) as valid. Otherwise, it drops it. Finally, if \(ID_i = ID_S\), then the node is the source node. In that case, the source node verifies the value \(hash_{K_{S,T}}(ID_S, ID_T, seq, RouteList_{j_i})\) and if it is valid, it stores the path \(p_j = (ID_S, RouteList_{j_i}, ID_T)\) as a valid path for communication with the target node \(T\).

A source node may communicate with the target node through the node-disjoint routing paths in three modes of operation. In the single operation mode, the source node \(S\) uses only the shortest communication path from the set \(M\). An alternative path is used if the path in use is broken. In the parallel operation mode the source node uses in parallel all the \(k\) valid node-disjoined paths it maintains towards \(T\). This is the most expensive operation mode and also the most secure, since at least \(k\) nodes must be simultaneously compromised for a successful DoS attack. Finally, in the mixed operation mode, communication is performed in the single operation mode and it switches to parallel when either of the two end-nodes requests this, e.g. when the exchanged information is critical or in case where a DoS is suspected.

#### 3.2.3. Route error algorithm

If a node \(n_i\) realizes during neighborhood authentication at time \(t+1\) that an established link with a neighboring node \(n_j\) during time \(t\) is now broken, then node \(n_i\) broadcasts a route error message for any active route coming through \(n_i\), that is affected due to the destruction of the link \((n_i, n_j)\). The route error message is digitally signed by the node \(n_i\). If the error messages are not signed, malicious nodes might flood the network with fake error messages even for routes that they do not participate in, and in this way disable communication. The route error message is of the form:
Any node that participates in the broken route marks the particular route as invalid and re-broadcasts the message until $S$ and $T$ are informed about the path breakage. According to the operation mode, an end node may restart the route request when: (i) a single path, (ii) a threshold number of paths or (iii) all existing paths from $S$ to $T$ are broken.

4. Security analysis

4.1. End-to-end route authentication

The route request is end-to-end authenticated with the security association $K_{S,T}$ that is exchanged. The keyed hash-value $\text{hash}_{K_{S,T}}(ID_S,ID_T,seq,ID_i,RouteList)$ included in the initial query, allows the target node to authenticate the request query. Note that in order to link the security association $K_{S,T}$ with the source node $S$, it may be required that $S$ also signs $K_{S,T}$ before encrypting it. However, the cost of public key cryptography is minimized since an Elliptic Curve Cryptosystem is used and only in an end-to-end basis. Moreover, if each pair of nodes shares a security association prior to the communication (which is possible for small-sized networks), the nodes do not require to exchange $K_{S,T}$ during the route request query.

4.2. Link-to-link route authentication

The links of a routing path are also authenticated indirectly, due to the neighborhood authentication phase of the protocol. Since each node $n_i$ periodically authenticates its neighborhood, each node may verify the authenticity of its own links inside any routing path that it participates in.

During the route request query propagation, each node $n_i$ accepts a request query, only if the last node identified in the partial $RouteList$ of the received route query message, is an authenticated neighbor of node $n_i$. Furthermore, during the route reply propagation, each node $n_i$ accepts and forwards a reply message, only if the two nodes identified before and after $ID_i$ in the routing path $p_j=(ID_S,RouteList_j,ID_T)$ of the reply message, are authenticated neighbors of $n_i$. Note that these verifications are very efficient since no cryptographic actions are required. Each node $n_i$ only verifies if the node identifiers in concern belong to its current neighborhood $N_i$. The use of the authenticated neighborhood is very efficient, since the cryptographic actions are performed once during a time period, regardless of the number of route request queries and route replies that pass through an intermediate node. Moreover, this common verification/authentication between neighboring nodes is less power consuming than having to reach the CA for common verification. This benefit is maximized for networks with relatively slow mobility. The duration of the time period is a system parameter that depends on how often the links between nodes are expected to break or to be established.

4.3. End-to-end route integrity

The routing path is also integrity protected during the route reply propagation in an end-to-end basis. Indeed, each route reply message includes a keyed hash-value $\text{hash}_{K_{S,T}}(ID_S,ID_T,seq,RouteList_j)$. Thus, if the routing path $p_j=(ID_S,RouteList_j,ID_T)$ has been altered, then the verification of the keyed hash-value will fail at node $S$ and the fake path will not be used.

4.4. Protection against malicious collaborating nodes

This is the basic security property that the protocol is designed for and it is achieved through multi-path routing. By using $k$ node-disjoint paths of communication, an adversary should compromise at least $k$ nodes – and more particularly at least one node in each route – in order to control the communication. According to the operation mode, SecMR offers different levels of protection. In parallel mode, the protocol is resilient against $k-1$ collaborating malicious nodes. In single operation mode the adversary can disrupt communication by compromising only the active path. The time required to activate an alternative path is still much less than in single-path routing protocols, but there are cases where such disruption may be critical. The protection of the mixed operation mode lies between the single and parallel mode and may be more efficient for practical applications.

Remark 2. This protocol does not fully protect from MIM and invisible node attacks. However, the SecMR protocol can be combined with lower-level multiple factor authentication mechanisms [6] to deal with these attacks.
Remark 3. Protection of the actual communication can be performed after the establishment of the routing paths with standard cryptographic techniques and secure message transmission protocols, e.g. [20]. Table 1 briefly presents the comparison issues that were discussed in this section.

5. Efficiency analysis

We analyze the performance characteristics and we present simulation results of the SecMR protocol with respect to other multipath protocols with comparable security properties, namely SecMR, Multipath [3] and SRP [19].

5.1. Message size

The size of the messages which are exchanged during the route discovery phase, is crucial for the efficient propagation of the route request query, especially in multipath routing protocols. If a route request query holds information concerning the intermediate nodes, then it faces the risk of oversized query messages. The SecMR protocol does not fall into this case since the size of the RouteList, the NextHop list and the ExcludeList, is growing in a controlled and predictable manner.

In particular, the maximum members that the RouteList is allowed to include is $R_{\text{max}}$, which represents the maximum allowed length of the discovered path as this is defined by $\text{hop}_{\text{max}}$ value. The NextHop list may include up to $N_{\text{max}} - 1$ members, where $N_{\text{max}}$ is the maximum number of neighbors that a node may have. The ExcludeList, in combination with the above, may include up to $1 + (N_{\text{max}} - 2) \times (R_{\text{max}} - 1)$ members.

Consider the scenario where a routing protocol is applied in a network where the nodes are scattered in clusters with at most $N_{\text{max}} + 1$ members. In order apply a routing protocol between all clusters, each two of these clusters should have at least one common node. In the worst case scenario, each pair of clusters will possess only one common node. Fig. 4 presents a typical ad hoc network that exhibits this kind of connectivity. By definition the SecMR does not allow the participation of nodes in more than one list. Thus, in the scenario where the clusters have more than one common neighbors the SecMR will prevent the ExcludeList to overcome its maximum size. Further on we will deal with the worst scenario as is the case where the ExcludeList may carry the maximum number of nodes.

Table 2 presents the growth of the message with regard to the members in the lists as this is produced by three comparable protocols, namely the SecMR, the Multipath of [3] and the SRP.

From the route request algorithm in Fig. 2 we see that

![Fig. 4. ExcludeList with maximum size.](image-url)
Table 2

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Maximum length of the lists</th>
<th>RouteList</th>
<th>NextHop List</th>
<th>ExcludeList</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecMR</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipath</td>
<td>1</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecMR</td>
<td>6 = (7 - 1)</td>
<td>6 = 0 + (7 - 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = 1 + (7 - 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipath</td>
<td>2</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecMR</td>
<td>6 = (7 - 1)</td>
<td>11 = 0 + 1 +</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(7 - 2) + (7 - 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipath</td>
<td>3</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecMR</td>
<td>(Nmax + 1), n &gt; 1</td>
<td>1 + (n - 1) *</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Nmax - 2), n &gt; 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipath</td>
<td>n</td>
<td>1 + n * (Nmax - 1),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>n &gt; 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ RouteList = RouteList + ID_i, \quad (2) \]
\[ NextHop = N_i - (N_i \cap RouteList) - (N_i \cap ExcludeList), \quad (3) \]
\[ ExcludeList = ExcludeList + (NextHop - ID_i). \quad (4) \]

By applying (2)–(4) in Table 2 for \( R_{\text{max}} = n + 1 \) we have

\[ RouteList \Rightarrow RouteList + ID_i = n + 1 = R_{\text{max}}, \]
\[ NextHop \Rightarrow N_i - (N_i \cap RouteList) - (N_i \cap ExcludeList) = N_{\text{max}} - (ID_i) - (N_{\text{max}} - ExcludeList) = N_{\text{max}} - 1, \]
\[ ExcludeList \Rightarrow ExcludeList + (NextHop - ID_i) \]
\[ \Rightarrow 1 + (n - 1) * (N_{\text{max}} - 2) + (NextHop - ID_i) \]
\[ \Rightarrow 1 + (n - 1) * (N_{\text{max}} - 2) + (N_{\text{max}} - 1 - ID_i) \]
\[ = 1 + (R_{\text{max}} - 1) * (N_{\text{max}} - 2). \quad (5) \]

Table 3 presents the growth of the messages with respect to the number of participating members inside the lists for the three multipath routing protocols. These are computed for a maximum hop length equal to 5 and for an average maximum node connectivity equal to 5.

SecMR manages to control the length of the generated lists of the route request query by controlling the overgrowth of the lists’ size. SRP produces small messages as it uses only one list, while Multipath faces the risk its messages to outgrow in size, as its NextHop list augments linearly with a higher factor then SecMR.

5.2. Performance evaluation

We implemented simulator within the NS-2 library. Our simulation modeled a network of 50 hosts placed randomly within a 1500 x 1000 m² area. Each node has a radio propagation range of 250 m and channel capacity was 2 Mb/s.

The nodes in the simulation move according to the “random way point” model. At the start of the simulation, each node waits for a pause time, then randomly selects and moves towards a destination with a speed uniformly lying between zero and the maximum speed. On reaching this destination it pauses again and repeats the above procedure till the end of the simulation. The minimum and maximum speeds are set to 0 and 20 m/s, respectively and pause times are 0, 5, 10, 20, 30 and 40 s. A pause time of 0 s corresponds to the continuous motion of the a node and a pause time of 40 s corresponds stationary node.

Ten traffic generators were developed to simulate constant bit rate (CBR) sources. Each source generates data packets continuously until the end of the simulation run. The sources and the destinations are randomly selected with uniform probabilities. The size of the data payload is 512 bytes. Each run is executed for 350 s of simulation time. We used the IEEE 802.11 Distributed Coordination Function (DCF) as the medium access control protocol. The destination of the traffic waits, if necessary, for 5 s until it assumes that all possible paths have been found, selects the node-disjoint ones and generate Reply messages concerning these paths. We generated various traffic scenarios by using the interarrival data packet time. For each traffic scenario, ten different movement patterns were used.

A free space propagation model with a threshold cutoff was used. In the radio model, we assumed the ability of a radio to lock onto a sufficiently strong

Table 3

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Maximum length of the lists</th>
<th>RouteList</th>
<th>NextHop List</th>
<th>ExcludeList</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecMR</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multipath</td>
<td>5</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
signal in the presence of interfering signals, i.e., radio capture.

In order to compare the performance of the three routing protocols we evaluated them with respect to the Average end-to-end delay or mean overall packet latency, Destination location time, Request Propagation Time, Drop percentage, Routing throughput [17].

Fig. 5 shows the average delay of the received data packets per data interarrival time and a pause time of 20 s. Both SRP and SecMR outperform Multipath even when the interarrival time is small, which depicts high traffic conditions. In both SRP and SecMR the number of generated messages during the route discovery process are kept in sufficiently low levels while the ones of Multipath tend to flood the network. This happens because in the Multipath protocol, each intermediate node forwards all the route requests that reaches it for a given source, destination and sequence number, while SRP forwards only the first and SecMR performs a selective forward with the use of the exclude list. This flooding of the network results in higher delay in the data packet delivery. Fig. 6, which presents the average delay of the received data packets to a network that transmits 100 data packets per second per with pause time, strengthens the above observations.

Fig. 7 presents the dropping percentage of the data packets in relation to interarrival times and a pause time of 20 s. All three protocols exhibit comparable performance, but SecMR and SRP manage to drop less packets, especially as the interarrival time is getting longer. The observed high dropping ratio, that all three protocols present is mainly due to the configuration of the simulation, namely the expiration time of the paths in the routing tables. Nevertheless, the performance pattern reveals the better performance of SecMR and SRP. Fig. 8 shows the dropping percentage of the data packets as this evolves in comparison to various pause times with a data interarrival of 0.25 s.

The number of packets that are correctly received by the destination node shown in Fig. 9. The performance of all three protocols tends to converge as the interarrival time gets longer, that is because in this
case the network is facing looser traffic conditions. As one can see SRP and SecMR manage to serve more packets in comparison to Multipath. That is mainly due to the fact that data packets in Multipath encounter higher delay during their propagation and higher dropping rate. All three protocols manage to maintain their behavior with regard to the message delivery ratio under various movability patterns, as shown in Fig. 10, which represent a data packet interarrival time of 0.01 s.

Fig. 11 presents the average total time that route request messages propagate through the network. In Multipath, as nodes are getting more and more stationary the total propagation time tends to get shorter until it reaches a minimum threshold. That does not seem to be the case for SecMR where the use of the ExcludeList prevents the request’s reception by nodes that have heard it sometime in the past. In the case of SRP the protocol benefits come from the fact that each route request is forwarded only once by each intermediate node.

Fig. 12 presents the average time it takes for a request message to reach its destination for the first time. If it is seen in comparison to Fig. 11, it is obvious that in Multipath the request message zombies into the network for longer time than in the other protocols, which causes a degradation to the network’s performance. In the secure Multipath, a route request travels for a longer time than in the other two protocols, as the request is being forwarded to all nodes in range, many of which will not be included into one of the discovered paths. The route request of the SRP propagates the request towards the destination faster than the other protocols, since it rejects any variant of a specific request. The route request of the SecMR has slightly longer living times than SRP. This is reasonable as it attempts to ensure the discovery of the complete set of existing node-disjoint paths. Furthermore, the SecMR makes sure that all its neighboring nodes have contributed to the route discovery, either by participating to the RouteList (i.e., to a routing path) or by avoiding to re-process the same thread of the query (i.e., by participating into the ExcludeList of the query thread). The above is, also, illustrated in Fig. 13, which presents
the total throughput derived from control messages averaged by the total number of nodes, for different pause times and with an interarrival time equal to 0.01 s. SRP produces less control overhead than the other two protocols, thus the reduced control throughput shown in Fig. 13. The control throughput of SecMR is slightly increased due to the selective forwarding of request messages that it performs. Multipath has the worst performance in comparison with the other two protocols, something that is due to the number of forwards that it performs.

6. Discussion

In this paper we proposed SecMR, a secure multipath routing protocol for ad hoc networks. A security concern of the SecMR is its dependance on the availability of a Certification Authority, during the node authentication phase, as an integral part for the routing mechanism. Although the use of a distributed PKI may reduce such problems, the implications of using the SecMR in periods of unavailability of a CA should be further investigated.

References


Rosa Mavropodi received her B.Sc. in Computer Science from the University of Piraeus, Greece in 1999 and since 2000 has been a PhD candidate there. Her main research interest relates to software engineering for telecommunication networks, distributed systems and architectures, intelligent networks, and performance evaluation of high speed networks.

Panayiotis Kotzanikolaou received his B.Sc. in Computer Science in 1998 and his PhD in 2003, both from the University of Piraeus, Greece. His research focuses on security and applied cryptography in the fields of mobile agents, distributed systems, intelligent networks, ad hoc networks and sensor networks.

Christos Douligeris received the Diploma in Electrical Engineering from the National Technical University of Athens in 1984 and the M.S., M.Phil. and Ph.D. degrees from Columbia University in 1985, 1987, 1990, respectively. He has held positions with the Department of Electrical and Computer Engineering at the University of Miami, where he reached the Greece. He has served in technical program committees of several conferences. His main technical interests lie in the areas of performance evaluation of high speed networks, neurocomputing in networking, resource allocation in wireless networks and information management, risk assessment and evaluation for emergency response operations. He was the guest editor of a special issue of the IEEE Communications Magazine on “Security for Telecommunication Networks” and he is preparing a book on “Network Security” to be published by IEEE Press/John Wiley. He is an editor of the IEEE Communications Letters, a technical editor of IEEE Network, and a technical editor of Computer Networks (Elsevier).