RESEARCH ARTICLE

Avoidance of misbehaving nodes in wireless mesh networks

Tal Grinshpoun¹ *, Amnon Meisels¹ and Eyal Felstaine²

¹ Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel
² Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

ABSTRACT

A wireless mesh network is a self organized set of nodes that are connected by wireless links. Communicating parties that are not in wireless range of each other relay packets via intermediate nodes. A common approach to wireless mesh routing is reactive routing, where a fixed path between the communicating endpoints is established on-demand when a new session is initiated. This paper proposes a distributed algorithm which guarantees service even when some wireless mesh nodes deliberately change, discard, or misroute data packets to disrupt service. When a misbehaving route is encountered, the proposed algorithm starts a process in which a “virtual” cost penalty is iteratively added to suspicious nodes and a new shortest route is derived until the disrupted path is replaced with one that avoids the misbehaving nodes. The algorithm enables proactive calculation of several alternative routes. The proactively calculated routes can be used to perform multipath routing that drastically enhances the robustness of the algorithm versus adversaries that dynamically change their behavior. Our algorithm can co-exist with common reactive wireless routing protocols. Furthermore, although every intermediate nodes may be malicious, the proposed algorithm does not impose costly authentication of messages from the participating intermediate nodes. This means that existing deployed infrastructures of wireless mesh nodes can be software-modified to work with the algorithm. We show that the proposed algorithm quickly converges to efficient alternative routes and present a bounded complexity for its time, communication, and computation overhead. Copyright © 2013 John Wiley & Sons, Ltd.

KEYWORDS
wireless routing; intrusion mitigation; security and system performance tradeoffs

*Correspondence
Tal Grinshpoun, Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
E-mail: grinshpo@cs.bgu.ac.il

1. INTRODUCTION

A wireless mesh network (WMN) [1] is a self organized, decentralized, relatively inexpensive, and moderately reliable network, made of nodes that are inter-connected by wireless links. Mesh router nodes form the backbone of the WMN and have minimal mobility, whereas mesh clients are mobile nodes that usually act as session endpoints. Mesh nodes forward packets on behalf of other nodes that may not be within direct wireless transmission range of their destinations.

The routing in WMNs is typically performed by protocols designed for mobile ad hoc networks (MANETs). Nevertheless, several routing protocols dedicated to WMNs were recently proposed. The wireless mesh networks routing protocol (MRP) [2] exploits the special structure of WMNs with a centralized structure to achieve improved performance. Ruiz and his colleagues proposed the use of multicast routing for WMNs [3], whereas other researchers attempted to utilize the use of multiple radios that are available in some WMNs [4,5]. Some of the proposed protocols for secure routing in WMNs strongly rely on the existence of a public-key infrastructure [6–8]. Although this assumption is reasonable in infrastructured WMNs, it is not always the case in hybrid WMNs, where some of the nodes may belong to different entities [1].

The limited mobility of mesh routers implies that routes between specific source and destination nodes will remain valid over long periods of time, unlike in the case of MANETs, where routes change rapidly. However, if a malicious, selfish, or simply overloaded node resides on such a long-living route, it may delay, change, drop, corrupt, or replay packets going through it and impact the entire session. WMNs typically employ low-cost outdoor devices that cannot be protected against removal, tampering, or replication by an adversary. Secure WMN routing protocols are difficult to design because the usage...
of low cost devices in WMNs leads to a scarcity in CPU resources, nodal storage space, and bandwidth. As a result, CPU-intensive, bandwidth-consuming, and cumbersome security mechanisms may consume excessive resources, leading to many new opportunities for denial-of-service (DoS) attacks through the routing protocol.

Following the depicted characteristics and vulnerabilities of WMNs, it is clearly essential to deal with the problem of misbehaving nodes. However, mitigating misbehavior might turn out very costly. Thus, methods that address this problem must be evaluated by several quality measures. The first measure is the convergence time of the network to a state of misbehavior-free communication. Another important measure is the continuity of communication. This measure refers to the amount of time that a communication session might be idle due to misbehavior. This can also be considered as the survivability of the network. The quality of the resulting misbehavior-free session is also of interest. An appropriate quality measure for a session is the length (number of hops) of the resulting route. The computational burden on the nodes, as well as the time and communication overhead are also to be taken under consideration when trying to deal with misbehavior. These issues are especially important in WMNs due to the mentioned characteristics of the mesh nodes. The relevant assessments are depicted in Sections 5 and 6.

A common approach for detecting misbehaving nodes is using watchdog agents that run on nodes and spy on peers in their vicinity [9]. The watchdog verifies that a packet received by an observed peer is indeed forwarded and is identical to the one received. This approach inevitably imposes a heavy communication burden on the WMN resources because every mesh node that acts as a watchdog must listen in a promiscuous mode and constantly exchange observations with collaborative watchdogs to gather and distribute reputation information [10,11]. Watchdogs might also be unable to detect a misbehaving node in the presence of packet collisions and amid a coalition of adversaries [9].

Similarly to watchdogs, collecting information on neighboring peers is the basis for a number of attempts to secure WMN routing. The enhanced secure routing protocol for wireless mesh networks (E-SRPM) [12] uses information of previous unsuccessful communications with neighboring nodes as one of the parameters for selecting routes. Two secure WMN routing protocols were proposed by Sen [13]. The first protocol relies on estimated reliability of routing paths by collecting information regarding the wireless links to one-hop neighbor nodes, whereas the second is a trust-based protocol that uses local observations for detecting selfish nodes.

Several solutions aim at protecting the route discovery process. The secure routing protocol [14] uses end-to-end authentication to prevent impersonation and replay attacks during route discovery. The Ariadne protocol [15] secures routing messages by requiring authentication for each intermediate node. Ariadne uses either pair-wise shared keys that require a high setup overhead, or Tesla [16], which is a broadcast authentication scheme that requires loose time synchronization. Similar security goals are achieved by using the secure efficient ad hoc distance vector routing protocol (SEAD) [17], which uses one-way hash chains to provide authentication, and by using the secure route discovery protocol (SRDP) [18], which relies on shared-key message authentication codes and public key signatures.

Routing misbehavior caused by a misbehaving node can be detected at the endpoints of a wireless communication session by applying a reliable authentication mechanism. Upon detecting such misbehavior, the on-demand secure Byzantine resilient routing protocol (ODSBR) [19] starts working in a probing mode, in which it attempts to pinpoint a culprit link. Subsequently, it penalizes the link and initiates a new route discovery process to bypass the misbehaving node.

Another method for bypassing misbehaving nodes is TIARA overlay routing [20], which uses a set of predefined “buddy” nodes. Whenever misbehavior is detected by the endpoints, the algorithm selects a buddy node and initiates a route discovery process to find the shortest path that goes through the buddy node. In case the new path fails at bypassing the misbehaving node, this process is repeated using another buddy node.

Both ODSBR and TIARA overlay routing assume that the network is not open. Specifically, that there exists a public-key infrastructure administered by a certificate authority. ODSBR requires this for its probing mode, whereas overlay routing must be familiar with the network to create the set of buddy nodes.

In this paper, we present a wireless mesh routing algorithm that bypasses misbehaving nodes. At its first stage, the algorithm effortlessly detects problematic routes (similar to ODSBR and TIARA). Then, it marks all nodes within these routes as suspects (of misbehavior). Finally, it finds new routes that avoid the misbehaving nodes. This is carried out by penalizing the costs of routes that go through suspects.

Although all the described schemes (the proposed algorithm, ODSBR, and TIARA) attempt to achieve the same goal of bypassing misbehaving nodes, the ODSBR protocol attempts to first pinpoint a culprit link. This computationally expensive task may be used by a sophisticated adversary to prevent ODSBR from ever initiating a new route discovery process. On the other hand, our proposed algorithm promptly initiates a new route discovery process. Moreover, the proposed algorithm enables proactive calculation of several alternative routes. These routes can be used to perform multipath routing that provides prompt discovery of misbehavior-free routes along with survivability and continuous communicability of the network even in the presence of multiple misbehaving nodes.

Another important contribution of the proposed algorithm is that the mesh routers only perform light computations, exclusive of any form of cryptographic operations. Nevertheless, the computational burden on the endpoints.
is similar to that in other approaches, such as ODSBR and TIARA overlay routing. Thus, the important advantages of the proposed algorithm are as follows:

- The algorithm does not expect mesh routers to sign or authenticate packets that pass through. This means that an increase in the processing power of mesh routers or usage of dedicated hardware is not required.
- The proposed solution relies only on authentication keys that reside at the endpoints and does not need a trusted key infrastructure at the mesh routers. Apart from being complicated and vulnerable, key infrastructure requires larger memory in mesh routers and consumes bandwidth for periodic maintenance. A trusted key infrastructure is still required for each pair of communicating mesh clients. However, this requirement is reasonable, because two communicating endpoints know and trust each other in the first place.
- Algorithms such as ODSBR and TIARA overlay routing require a closed network with a certified authority. However, the proposed algorithm can be applied in dynamic and more realistic networks that lack any form of centralized authority.
- Multipath routing can be enabled with the new algorithm. This results in effective route discovery, survivability, robustness, and continuous communicability of the network even in the presence of several misbehaving nodes.
- The communications overhead and convergence time of the proposed algorithm are shown to be bounded and competitive. It is comparable with the overhead that is required for finding a route in the absence of misbehaving nodes.
- In simulation experiments it was found that the lengths of the alternative routes are close to optimal.

The first two properties essentially mean that unlike ODSBR, it is possible to incorporate the proposed algorithm into state of the art WMNs without imposing a significant change to the existing routing hardware, which is impractical for already deployed WMNs.

The rest of this paper is organized as follows. Section 2 provides an overview of the proposed algorithm, and Section 3 presents several extensions. The resilience of our scheme to various attacks is presented in Section 4, and an analysis of the overhead complexity is provided in Section 5. Section 6 includes an experimental evaluation of the proposed algorithm. The conclusions are presented in Section 7.

2. ALGORITHM OVERVIEW

The avoidance of misbehaving nodes (AVOMIN) algorithm works on top of an underlying source routing protocol that performs a distributed version of shortest path routing (DSR) [21]. The DSR protocol is described in Section 2.1. For simplicity, we assume that all the links in the communication graph are bidirectional and that the routing protocol uses the traditional number-of-hops metric, that is, a uniform weight of 1 is given to all the links in the network. When the entire network behaves properly, AVOMIN does not alter the shortest path routes detected by the underlying routing protocol.

The AVOMIN algorithm comes into action when a node along an active session route starts misbehaving. The existence of misbehaving nodes is discovered by a reliable authentication mechanism that is deployed at the edges of every wireless communication session. Hence, the endpoints identify cases of forged, dropped, duplicated, or altered packets. After the number of these occurrences surpasses a predefined threshold, the session is considered to be affected by node misbehavior. The source node defines this route as a suspected route, and the set of nodes along the suspected route (not including the source and destination nodes) are defined as the suspect list.

In an attempt to avoid the misbehaving node, the algorithm temporarily and gradually assigns an artificial weight to all the nodes that belong to the suspect list. Namely, the weights of the suspected nodes are increased by a predetermined $\delta$ value.

The routing algorithm is likely to discover a new path that bypasses the misbehaving node. However, in the event that an increase in weight does not yield a route that is free of misbehaving nodes, the secure layer at the endpoints continues to detect suspicious behavior. The iterative process of increasing $\delta$ to the weights of the suspected nodes continues until a route free of misbehaving nodes is discovered. Throughout the iterative process, the same suspect list (which was derived from the original suspected route) is used. This enables the proactive routes calculation extension that is described in Section 3.1.

The artificial increase in weight is performed per sender and does not impact sessions from any other senders. Namely, the nodes do not increase these weights when forwarding packets belonging to sessions of other senders. This separation between the real weights and the artificially increased weights disallows potential blackmail attacks [15], in which a malicious node tries to impair the overall throughput of the network by falsely accusing well-behaving nodes of misbehavior. Consequently, a malicious node that tries such an attack will only harm its own sessions, which is clearly useless.

The sender may also choose to periodically reset its suspect list to allow the resurrection of nodes that were only temporally misbehaving. Such resuscitation is inherently achieved when the multipath routing extension is used (see Section 3.3).

As an example, consider the wireless mesh network depicted in Figure 1. The network is composed of several wireless mesh routers. A link between two mesh routers indicates that the receiver can hear the sender and thus the link is capable of relaying packets.

Assume that endpoint devices $S$ (source) and $D$ (destination) establish a session. As evidenced by the links in Figure 1, the shortest path is $\{S,K,J,I,D\}$ (the weight of the path equals the number of hops which is 4). Assum-
needed. This is carried out by adding another iteration of the route discovery process (which contains the misbehaving node) and distributes it throughout the network as part of a new route discovery process.

During the new route discovery process, the weights of the nodes in the suspect list \( \{K, J, I\} \) are increased (penalized) by \( \delta \), which is equal to 0.6 in this example. Figure 2 presents the same network after the new route discovery process is completed.

As can be seen in Figure 2, penalizing \( \delta \)'s were added to the three nodes of the original route – \( K, J, \) and \( I \). Consequently, the new weight of route \( \{S, K, J, I, D\} \) is now 5.8, which is not the shortest path anymore. The new “shortest” path is \( \{S, O, N, M, I, D\} \), and its weight is 5.6.

In case the real misbehaving node is \( K \) or \( J \), the outcome of the route discovery process is a successful bypassing and avoidance of the misbehaving node. However, if node \( I \) is the misbehaving node, the endpoints \( S \) and \( D \) are likely to observe misbehaving activity along the new “shortest” path (which contains the misbehaving node \( I \)). Consequently, an additional iteration of the route discovery process is needed. This is carried out by adding another \( \delta \) to all the nodes in the suspect list, increasing their penalty to 1.2. After the second iteration, the new weight of route \( \{S, O, N, M, I, D\} \) is 6.2, which is no longer the shortest path. The new shortest path is any 6-hop route that contains no suspected nodes, for example \( \{S, L, H, G, F, E, D\} \), and its weight is 6.0.

### 2.1. Dynamic source routing

The dynamic source routing protocol (DSR) [21] is a widely used reactive on-demand routing protocol for wireless networks and specifically WMNs. It is based on source routing, in which every packet holds the entire route path consisting of the addresses of all the nodes in the route. DSR is comprised of two mechanisms that work together and allow the discovery and maintenance of source routes in DSR. Route discovery is the mechanism by which a source node \( S \) that aims at sending a packet to a destination node \( D \) obtains a source route to \( D \). Route maintenance is the mechanism by which node \( S \) is able to detect, while using a source route to \( D \), if the network topology has changed in such a way that it can no longer use its route to \( D \). In such a case, node \( S \) can invoke a new route discovery process.

The route discovery mechanism is illustrated in Figure 3. When a source node \( S \) wishes to communicate with a destination node \( D \) without knowing any paths to \( D \), \( S \) initiates a route discovery process. \( S \) starts by transmitting a route request (RREQ) message that contains the address of \( D \) to all its neighbors. The neighbors in their turn append their own addresses to the first RREQ they receive and retransmit it. Each node discards multiple copies of the same RREQ. Consequently, the RREQ message is flooded throughout the network until it reaches its destination. Upon receiving the RREQ, the destination node \( D \) sends a route reply (RREP) message back to the source node \( S \). Assuming that the links are bidirectional, the RREP can be sent back to \( S \) in exactly the reverse way that the corresponding RREQ reached \( D \). Otherwise, another flooding phase is initiated by \( D \) with a message containing the path from \( S \) to \( D \).

The route maintenance mechanism operates entirely on-demand. It does not use any link status sensing or neighbor detection messages, and does not rely on such functions from any underlying layer of the network. Instead, a link break is detected only when a node forwarding a packet to the next node in the route path notices that its packet did not reach the next node. This method of detection is possible by using a confirmation that a packet was received.
Avoidance of misbehaving nodes in wireless mesh networks

T. Grinshpoun, A. Meisels and E. Felstaine

Figure 2. Example network with suspected nodes (circled).

Figure 3. Route discovery. (a) Node S sends a RREQ to find a path to node D. The route request (RREQ) is flooded throughout the network. (b) Node D sends back to S a route reply by using the bidirectional path contained in one of the RREQ messages that reached it.

2.2. Route discovery in avoidance of misbehaving nodes

The choice of using DSR as the underlying routing protocol is not accidental. Only a source routing protocol such as DSR inherently provides a suspect list that the AVOMIN algorithm can use. In other protocols such as optimized link state routing (OLSR) [23], destination-sequenced distance vector routing (DSDV) [24], and ad-hoc on-demand distance vector routing (AODV) [25], the routing protocol uses local information from each intermediate node. When using such protocols, node misbehavior can be detected with the help of watchdog agents [9]. However as discussed in the introduction, such a routine heavily burdens the intermediate nodes and may not always detect misbehavior. As a result, AVOMIN’s endpoint-to-endpoint approach would lose its appeal. Consequently, DSR is used as the underlying routing protocol henceforward.

When the network behaves properly, new routes are discovered using the original route discovery mechanism of DSR. The AVOMIN algorithm comes into action when misbehavior is detected during some session. At that point, the source node of the session creates a suspect list, and initiates a new route discovery process (Algorithm 1). However, contrary to the DSR route discovery, the RREQ messages that are flooded throughout the network also hold the suspect list. The part in the RREQ message holding the suspect list and the iteration number is digitally signed by the source node to deal with possible attacks on the protocol (Section 4). Notice, that sessions that are not affected by misbehavior still use the original DSR route discovery mechanism.

An intermediate node which receives a RREQ message that was originated by S first calculates the route’s weight. In the DSR protocol the weight calculation is straightforward, since the RREQ message received at a given node contains the route from the source to that node. However in AVOMIN, penalizing δ’s should be added for suspected nodes that belong to the current route. This is done by multiplying the number of suspected nodes in the route by the
Avoidance of misbehaving nodes in wireless mesh networks

Algorithm 1 Basic AVOMIN: misbehavior detection (node $i$)
upon detecting misbehavior on route $\text{ROUTE}_{s,d}$:
\begin{enumerate}
\item if this is the first misbehavior detected on $\text{ROUTE}_{s,d}$ then
\begin{enumerate}
\item create a suspect list $S\text{LIST} = \{x|x \in \text{ROUTE}_{s,d}, x \neq s, x \neq d\}$
\item iteration $\leftarrow 1$
\end{enumerate}
\item else
\begin{enumerate}
\item iteration $\leftarrow$ iteration + 1
\end{enumerate}
\end{enumerate}
end if
\end{enumerate}
\begin{enumerate}
\item set $\text{ROUTE} \leftarrow \{s\}$
\item sign $S\text{LIST}$ and iteration
\item transmit $(\text{RREQ}, s, d, \text{msg}\_id, \text{ROUTE}, S\text{LIST}, \text{iteration})$
\end{enumerate}

Algorithm 2 Basic AVOMIN: route request (node $i$)
upon receiving $(\text{RREQ}, s, d, \text{msg}\_id, \text{ROUTE}, S\text{LIST}, \text{iteration})$:
\begin{enumerate}
\item if $i = d$ then
\begin{enumerate}
\item verify signature of $S\text{LIST}$ and iteration according to public key of $s$ and exit if verification failed
\end{enumerate}
\end{enumerate}
end if
\begin{enumerate}
\item if $\text{RREQ}$ table does not have an entry $\text{table}\_\text{ent}$ with $s$ and $\text{msg}\_id$ then
\begin{enumerate}
\item create entry $\text{table}\_\text{ent}$ in $\text{RREQ}$ table
\item $\text{table}\_\text{ent}\_\text{weight} \leftarrow \infty$
\item $\text{table}\_\text{ent}\_\text{ROUTE} \leftarrow \emptyset$
\end{enumerate}
\end{enumerate}
end if
\begin{enumerate}
\item $\text{add} \leftarrow (S\text{LIST} \cap \text{ROUTE})$
\item new weight $\leftarrow |\text{ROUTE}| + \delta \cdot \text{add} \cdot \text{iteration}$
\item if new weight $< \text{table}\_\text{ent}\_\text{weight}$ then
\begin{enumerate}
\item $\text{table}\_\text{ent}\_\text{weight} \leftarrow \text{new weight}$
\item $\text{table}\_\text{ent}\_\text{ROUTE} \leftarrow \text{ROUTE} \cup \{i\}$
\end{enumerate}
\item wait (see equation 1 in section 3.2)
\item if $i = d$ then
\begin{enumerate}
\item sign and transmit $(\text{RREP}, d, s, \text{msg}\_id, \text{table}\_\text{ent}\_\text{ROUTE})$
\end{enumerate}
\item else
\begin{enumerate}
\item transmit $\text{< RREQ, s, d, msg_id, table_ent.ROUTE, SLIST, iteration >}$
\end{enumerate}
\end{enumerate}
end if
end if

After receiving RREQ messages from several of its neighbors, the intermediate node prunes all the ones that do not have the minimal weight. It then appends its identity to the route with minimal weight (according to its own calculations), and resends the RREQ message. Notice, that to prune non-minimal routes, the node has to wait for some time before it forwards the RREQ message. Section 3.2 discusses the optimal waiting time.

Once a RREQ message reaches the destination node, the latter creates a RREP message containing the received route. It then digitally signs the RREP message (see Section 4) and sends it in the reverse path of the route request (Algorithm 3). This can be carried out under the assumption that the links are bidirectional.

It seems very tempting to try and pinpoint the misbehaving nodes once a suspect list is available. The ODSBR protocol [19] follows this plan as it enters its probing mode. Our analysis (Section 5) and experimental evaluation (Section 6) show that such an approach is not necessarily worthwhile. The fact that in AVOMIN only the endpoints need to sign and verify the messages enables the use of already deployed WMNs without any form of trusted key infrastructure at the mesh routers.

Algorithm 3 Basic AVOMIN: route reply (node $i$)
upon receiving $(\text{RREP}, d, s, \text{msg}\_id, \text{ROUTE})$:
\begin{enumerate}
\item if has not yet received this message and $i \in \text{ROUTE}$ then
\begin{enumerate}
\item if $i = s$ then
\begin{enumerate}
\item verify signature according to public key of $d$ and exit if verification failed
\end{enumerate}
\item update routing table with $\text{ROUTE}$
\end{enumerate}
\item else
\begin{enumerate}
\item retransmit $(\text{RREP}, d, s, \text{msg}\_id, \text{ROUTE})$
\end{enumerate}
\end{enumerate}
end if
end if

To demonstrate the AVOMIN route discovery process, consider the intermediate node $F$ depicted in Figure 2. Node $F$ receives a RREQ message from node $J$, which contains the route $\{S, K, J\}$. Because both $K$ and $J$ are suspected nodes, node $F$ adds two $\delta$’s to the number of hops, which is $2$ excluding the $(J, F)$ hop. Given $\delta = 0.6$, the resulting weight is $3.2$. However, this message is pruned, because $F$ also receives an RREQ message from $G$, which consists of $3$ hops and no suspected nodes ($\{S, L, H, G\}$). Node $F$ appends its identity to the partial route and resends the updated RREQ message $\{S, L, H, G, F\}$. Eventually an RREQ message reaches the destination node $D$, say $\{S, L, H, G, F, E, D\}$. Node $D$ creates a RREP message containing the received route and sends it in the reverse path $\{D, E, F, G, H, L, S\}$.

3. PROTOCOL EXTENSIONS

3.1. Proactive routes calculation

The route discovery process discussed in Section 2.2 imposes the flooding of RREQ messages throughout the network. When several iterations are needed until a route free of misbehaving nodes is found, the implied messaging overhead can become substantial. To mitigate such an overhead we propose the proactive routes calculation method.

The idea is to proactively calculate the routes resulting from future iterations. This can be achieved by extending the initial RREQ message to include the shortest routes for several iterations in advance (Algorithm 4). An intermediate node gathers all the routes it receives and chooses the shortest route for every iteration (Algorithm 5). The node therefore forwards an RREQ message consisting of the shortest route for the first iteration, the shortest route for the second iteration, ..., and the shortest route for the $k$th iteration, where $k$ is a predefined number of proactively calculated iterations. The route reply message consists of the $k$ resulting routes (Algorithm 6). The shortest routes from all the iterations can be proactively
calculated because the same suspect list is used throughout the iterative process, and only the iteration number changes (see Algorithm 1).

**Algorithm 4 AVOMIN with proactive routes calculation: misbehavior detection (node $s$)**

```
upon detecting misbehavior on route $\text{ROUTE}_{s,d}$:
  if $\mathcal{A}_c$ cache not empty then
    extract shortest $\text{ROUTE}$ from $\mathcal{A}_c$ cache
    update routing table with $\text{ROUTE}$
  else
    create a suspect list $S\text{LIST} = \{x | x \in \text{ROUTE}_{s,d}, x \neq s, x \neq d\}$
    for each iteration $1 \ldots k$ do
      $\text{ROUTE}_{\text{iteration}} \leftarrow \{s\}$
      sign $S\text{LIST}$
      transmit ($\text{RREQ}, s, d, \text{msg.id}, \text{ROUTE}_1 \ldots \text{ROUTE}_k, S\text{LIST}$)
  end if
```

**Algorithm 5 AVOMIN with proactive routes calculation: route request (node $s$)**

```
upon receiving ($\text{RREQ}, s, d, \text{msg.id}, \text{ROUTE}_1 \ldots \text{ROUTE}_k, S\text{LIST}$):
  if $i = d$ then
    verify signature of $S\text{LIST}$ according to public key of $s$ and exit
  else if verification failed then
    end if
  if $\text{RREQ}$ table does not have an entry $\text{table.ent}$ with $s$ and $\text{msg.id}$ then
    create entry $\text{table.ent}$ in $\text{RREQ}$ table
    for iteration $1 \ldots k$ do
      $\text{table.ent.ROUTE}_{\text{iteration}} \leftarrow \infty$
      $\text{table.ent.ROUTE}_{\text{iteration}} \leftarrow \emptyset$
    end for
  end if

  updated $\leftarrow$ false
  for iteration $1 \ldots k$ do
    new_weight $\leftarrow |\text{ROUTE}_{\text{iteration}}| + \delta \cdot \text{add iteration}
    if new_weight < $\text{table.ent.weight}_{\text{iteration}}$ then
      $\text{table.ent.weight}_{\text{iteration}} \leftarrow \text{new weight}$
      $\text{table.ent.ROUTE}_{\text{iteration}} \leftarrow \text{ROUTE}_{\text{iteration}} \cup \{i\}$
      updated $\leftarrow$ true
    end if
  end for
  if updated $= true$ then
    wait (see equation 2 in section 3.2)
    if $i = d$ then
      sign and transmit ($\text{RREP}, d, s, \text{msg.id}$,
      $\text{table.ent.ROUTE}_1 \ldots \text{table.ent.ROUTE}_k$)
      else
        transmit ($\text{RREQ}, s, d, \text{msg.id}$,
        $\text{table.ent.ROUTE}_1 \ldots \text{table.ent.ROUTE}_k, S\text{LIST}$)
      end if
  end if
```

At the end of the proactive route calculation, the source node has all the necessary data to repenalize the suspected nodes and find alternative routes, without having to flood the network all over again. The source node can also skip redundant iterations, where the resulting route does not change.

**Table I.** Results of proactive routes calculation for the network depicted in Figure 2.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Penalty</th>
<th>Shortest route</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.0</td>
<td>${S, K, J, I, D}$</td>
<td>4.0</td>
</tr>
<tr>
<td>First iteration</td>
<td>0.6</td>
<td>${S, O, N, M, I, D}$</td>
<td>5.6</td>
</tr>
<tr>
<td>Second iteration</td>
<td>1.2</td>
<td>${S, L, H, G, F, E, D}$</td>
<td>6.0</td>
</tr>
<tr>
<td>Third iteration</td>
<td>1.8</td>
<td>${S, L, H, G, F, E, D}$</td>
<td>6.0</td>
</tr>
</tbody>
</table>

For example, consider the network depicted in Figure 2 with $\delta = 0.6$ and $k = 3$. The source node $S$ will receive the shortest route for every iteration as listed in Table I. Notice that the third iteration is redundant, because it obtains exactly the same route as the second iteration. Such a redundant iteration will be skipped by the source node.

### 3.2. Route request waiting mechanism

The route discovery process in AVOMIN consists of two phases – RREQ and RREP. The RREQ messages are flooded throughout the network. If no misbehavior is detected, the AVOMIN RREQ is identical to the DSR RREQ [21] and consists of $V$ transmissions (all the nodes transmit the RREQ after they have received it for the first time). However, when the RREQ contains a suspect list, some of the intermediate nodes must transmit the RREQ several times. This can happen when a node first receives a RREQ that has passed through several suspects, and later on it receives a second copy of the RREQ that has passed through a longer path but with less (or no) suspects, which has a lower weight than the first path. In the worst case, the RREQ is transmitted an exponential number of times [26].

To prevent such redundant transmissions, we suggest a waiting mechanism in which an intermediate node waits before forwarding a RREQ message. Each node transmits the RREQ message only once, cutting down the overall number of RREQ transmissions in the network to $V$, as in DSR [21].

The most important aspect of the RREQ waiting mechanism is deciding on the amount of time that an intermediate node should wait before forwarding a RREQ. Consider the weight of the first copy of the RREQ message that an intermediate node $I$ receives as $W_{\text{ROUTE}}$, which is composed of
the length of the route with the addition of δ’s associated with suspected nodes. The receiving node needs to wait for copies of the RREQ with a lower cost than that of the first copy it has received. Knowing the value of \( W_{\text{ROUTE}} \), the node under discussion can derive the length of the longest possible route that it must wait for, which is likely to be a route that contains no suspects. Consequently, the maximal possible length of a route is bounded to \( [W_{\text{ROUTE}}] \).

We denote \( T_{\text{hop}} \) as the maximal time it takes for a RREQ message to advance a single hop. \( T_{\text{hop}} \) includes both the processing time within an intermediate node and the transmission time of the message before it reaches the next node. We can use such an upper bound, because a node or a link that operates slower than this bound is considered malfunctioned and is temporarily removed from the network.

By combining the maximal route length and the maximal hop time, we derive that the maximal time that it might take the lowest cost RREQ to reach node \( I \) is \( [W_{\text{ROUTE}}] \cdot T_{\text{hop}} \).

However, node \( I \) is not aware of the exact time in which the RREQ was originally sent by the source node \( S \). This problem is solved by applying loose time synchronization between nodes. Let \( \Delta \) be the maximum time synchronization error between any two nodes; the value \( \Delta \) must be known by all nodes. Consequently, each intermediate node has to wait an extra \( \Delta \). These additional \( \Delta \) values accumulate in every hop. Thus, the maximal waiting time is as follows:

\[
T_{\text{wait}} = TIME_{\text{msg}} + [W_{\text{ROUTE}}] \cdot (T_{\text{hop}} + \Delta) - TIME_I \tag{1}
\]

where \( TIME_{\text{msg}} \) is the time when the source node \( S \) sent the original RREQ message and \( TIME_I \) is the current time at node \( I \).

The waiting time for the destination node \( D \) incorporates the waiting times for all intermediate nodes because the mutual synchronization with the source node \( S \) \((TIME_{\text{msg}})\). Consequently, \( T_{\text{wait}} \) for node \( D \) reflects the total time overhead incurred by the waiting mechanism.

Note that to apply the RREQ waiting mechanism, the \( TIME_{\text{msg}} \) field has to be added to the RREQ messages. It should be digitally signed along with the \( S_{\text{LIST}} \) field to ensure that an adversary cannot change its value without being noticed.

The RREQ waiting mechanism can also be applied along with the proactive routes calculation method (Section 3.1). It is important to notice that the longest possible route grows as the penalty to each suspected node grows. Consequently, the longest possible route in a proactive route calculation is found in the \( k' \)th (last) iteration. Thus, we can derive the maximal waiting time with proactive routes calculation:

\[
T_{\text{wait, pro}} = TIME_{\text{msg}} + [W_{\text{ROUTE}}] \cdot (T_{\text{hop}} + \Delta) - TIME_I \tag{2}
\]

3.3. Multipath routing

Survivability is a vital property of wireless networks. It means that even amid adversaries, the communication in the network should remain continuously operative. AVOMIN with multipath routing enables the endpoints to continuously communicate throughout a timely convergence phase even amid an attack of several adversaries, thus drastically improving the network’s survivability.

The AVOMIN multipath routing algorithm, which is deployed by the endpoints, is described as follows:

1. Run a proactive routes calculation (Section 3.1) to obtain up to \( K \) different routes between the source node \( S \) and the destination node \( D \). We refer to each of the attained routes as a subflow. This step could be executed in advance before any misbehavior has occurred in order to reduce the setup time of AVOMIN. In case the dropped-packet ratio (DPR) of the shortest path surpasses a predefined value, the session is considered to be carried by misbehaving nodes, and the algorithm advances to stage 2.

2. Assign a fitness score to each subflow. The initial score depends on the length of the subflow – shorter subflows (in terms of number of hops) gain higher initial scores (see Formula 3).

3. Distribute the packets that belong to the session among the different subflows according to a predefined policy. Subflows with a higher fitness score will relay a larger portion of the packets.

4. Keep track of the DPR of each subflow and use it to update the fitness score of the subflow, which is calculated using the following formula:

\[
\text{FITNESS}_{\text{subflow}} = \frac{(1 - \text{DPR})^2}{\text{length}} \tag{3}
\]

Then, go back to stage 3 and distribute the packets among the subflows by using the new fitness scores. Throughout the session, any dropped packet is retransmitted by node \( S \) through a subflow different from the one that has initially dropped the packet. This is carried out to reduce the chances of a dropped retransmitted packet, a property which is important to the survivability of the network.

For example, consider the network depicted in Figure 4 and the proactively calculated route list presented in Table 1. In this simple example, the subflows (denominated as \( SF \)) are \{S, K, I, D\} \((SF_0)\), which is also the original shortest path, \{S, O, N, M, I, D\} \((SF_1)\), and \{S, L, H, G, F, E, D\} \((SF_2)\). These subflows are attained in advance (see Section 3.1) even before any misbehavior has occurred to reduce the setup time in the event that misbehavior does eventually occur. Nevertheless, in case no misbehavior is detected in \( SF_0 \), the other subflows are not used. However, in case the DPR, the percentage of the packets that do not reach the destination, surpasses a prede-
Avoidance of misbehaving nodes in wireless mesh networks

T. Grinshpoun, A. Meisels and E. Felstaine

Figure 4. Example network with multipath routing.

fined τ threshold, the protocol begins to work in multipath routing mode.

Consider I to be an adversary that maliciously desires to hinder the communication within the network. Assume that at some point the adversary begins dropping half of the packets that it receives. Consequently, the DPR of subflow SF₀ surpasses the τ threshold (in this example τ = 20%), and the protocol begins to work in multipath routing mode. At this stage only SF₀ has DPR data. Hence, the initial fitness score of subflows SF₁ and SF₂ relies solely on their lengths, whereas SF₀ also relies on the measured DPR:

\[
FITNESS_{SF₀} = \frac{(1 - 0.5)^2}{4} = 0.0625
\]
\[
FITNESS_{SF₁} = \frac{(1 - 0)^2}{5} = 0.2
\]
\[
FITNESS_{SF₂} = \frac{(1 - 0)^2}{6} = 0.1667
\]

The aim in using a square of (1-DPR) in the formula, is to drastically reduce the fitness scores whenever misbehavior occurs.

Now, the packets need to be distributed on the basis of the fitness scores of the subflows. In this example, the share of packets that are routed through a subflow is calculated by the formula:

\[
\%PACKETS_{SFj} = \frac{(FITNESS_{SFj})^2}{\sum_{j=1}^{k} (FITNESS_{SFj})^2}
\]

Thus, at this stage, 5.4% of the packets would be routed using SF₀, 55.8% would be routed using SF₁, and 38.8% would be routed using SF₂. Because node I resides on SF₁, and because it continues to drop half of the packets that it receives, the DPR of subflow SF₁ also reaches 50%. Thus, its new fitness score is as follows:

\[
FITNESS_{SF₁} = \frac{(1 - 0.5)^2}{5} = 0.05
\]

Consequently, at this stage 11.4% of the packets would be routed using SF₀, 7.3% would be routed using SF₁, and 81.3% would be routed using SF₂. Hence, 81.3% of the packets in the session would bypass the malicious node, and 90.4% would not be dropped. If node I decides to drop a larger share of the packets, it would just cause the fitness scores of the subflows that it resides on (SF₀ and SF₁) to further drop. Thus, the adversary enters a “catch-22” state, in which it must forward packets correctly to prevent its subflow’s fitness score from dropping and thereby cutting him out of the data stream. The continuous update of fitness scores allows the algorithm to effortlessly adapt to changes in the adversarial behavior. It also allows the resuscitation of subflows. For example, if node I was only temporarily misbehaving as a result of some malfunction, the fitness scores of subflows SF₀ and SF₁ would eventually increase, and a large share of the packets would be passed through them.

If all the subflows experience misbehavior then the aggregated dropped-packet ratio of the multipath session may surpass the threshold. This obviously requires finding new subflows by applying an additional proactive routes calculation. This time, the suspect list used for the calculation consists of the nodes in all the failed subflows (SF₀, SF₁, and SF₂ in our example).

The presented example shows how AVOMIN multipath routing copes with sophisticated adversaries that selectively drop packets. Other examples include a consistent adversary or a totally malfunctioned node, both of which consistently drop all the received packets. In such cases, the respective subflows suffer from 100% DPR, thus no
traffic would be passed through them. However, the endpoints would still periodically relay a very small amount of packets through these subflows to allow their resuscitation.

Naturally, when using multipath routing, not all the packets arrive at their destination appropriately. There are commonly two ways to deal with this problem. The first one is retransmitting such packets. A variation of this approach that uses alternate subflows is used in the AVOMIN multipath routing algorithm (stage 4). The other approach is to replicate packets in advance and to distribute them between different paths with or without coding [27–29]. The assumption in such an approach is that the different paths are disjoint and each has a similar chance of packet dropping. For this reason, solutions that follow this approach (e.g., erasure coding [27]) are not fitted for working with AVOMIN’s subflows. Moreover, because the subflows in AVOMIN are computed especially to circumvent the misbehaving nodes, any solution that proactively replicates packets adds a redundant overhead and is therefore considered an overkill.

One may notice that the computation of a subflow’s fitness (Formula 3) resembles the expected transmission count metric (ETX) for finding high-throughput routes [30]. The ETX of a link considers the delivery ratio (1-DPR), whereas the ETX of an entire route is a summation of the ETX for each link in that route. It is not surprising that both the ETX metric and Formula 3 consider the DPR along with the length of the route, because it was shown that only when considering both these factors, one can appropriately measure a route’s throughput [30].

4. RESILIENCE TO ATTACKS

The AVOMIN algorithm is resilient to many forms of attacks. Next, we introduce various known attacks against routing protocols and describe how the AVOMIN algorithm handles these attacks.

A black hole attack is a basic attack in which the adversary drops entirely or selectively data packets, while still participating in the routing protocol. We also consider attacks where an adversary alters, corrupts, duplicates, or forges data packets as cases of the black hole attack.

The AVOMIN algorithm uses a reliable authentication mechanism that is deployed at the endpoints to detect packet losses (including packet alteration, corruption, duplication, and forging). When the number of lost packets becomes higher than a threshold r, a black hole attack is detected. At that point, the AVOMIN algorithm attempts to find a new route that bypasses the adversary by penalizing suspected nodes. The algorithm will continue with the process of penalizing suspected nodes until it finds a route free of misbehavior.

In a flood rushing attack [31] the adversary responds quickly to RREQ messages and promptly forwards them. This attack exploits the fact that many wireless routing protocols (for example, DSR [21]) forward only the first copy of each RREQ message they receive. This way, an adversary may control many paths in the network.

The flood rushing attack cannot occur in AVOMIN, because the algorithm retransmits a RREQ message if it has a lower metric than a previously arrived copy of the same RREQ (due to penalized nodes). In case an adversary that is not suspected of misbehavior uses flood rushing to control a path and then begins dropping packets (black hole attack), the AVOMIN algorithm will mark it as a suspect and penalize it. Moreover, applying the waiting mechanism (Section 3.2) completely mitigates rushing attacks.

A blackmail attack [15] is when an adversary accuses a good node of misbehaving. This attack was originally discussed as an attack against the watchdog mechanism [9]. However, blackmailing may be applied against any protocol that maintains knowledge about past misbehavior, such as the penalized nodes in AVOMIN. Nevertheless, the AVOMIN algorithm is resilient to blackmail attacks, because knowledge about a past misbehavior is maintained in the context of the sender and only the sender can create suspect lists. Consequently, an adversary can only falsely accuse nodes in the context of its own sessions, which is clearly useless.

In the wormhole attack [32,33], an adversary, or potentially multiple colluding adversaries, surreptitiously relay packets between distant locations. This can give a node the impression that it is the neighbor of another node that is actually far away. By faking links between distant nodes, adversaries may be able to manipulate nodes to send traffic through them.

The AVOMIN algorithm cannot detect a wormhole, but it can mitigate its effect whenever packets going through the wormhole are being dropped (black hole attack). In this case, the participants of the wormhole would be marked as suspects, and eventually the AVOMIN algorithm would find a route that bypasses the wormhole (if such route exists).

A short route attack is an attack on the route discovery mechanism. An adversary that wishes to attack as many sessions as possible would like to make routes that go through it seem shorter in the route discovery phase. To do this the adversary needs to resend a RREQ with a route shorter than the shortest route it received. Assuming that eventually the destination node D chooses this route and sends it back with a RREP, the adversary cannot change the route on the way back because the RREP message is digitally signed by node D. If it does try to change it, the RREP message will fail the verification when it finally reaches the source node S. Consequently, the adversary will have to either send the RREP message as it is or drop it. Because the message contains a route back from the adversary to the source node S that is shorter than the real shortest route, the message will fail to reach node S, and node S will not choose it.

Another potential threat is when several colluding adversaries cunningly time their offense, by attacking the network one at a time, rather than all at once. Such an offense may have severe consequences on iterative proto-
cols, such as TIARA overlay routing and ODSBR, which deal with each problem at a time. However, AVOMIN with multipath routing completely mitigates the added effect of considerate timing by the adversaries, because several routes are used at the same time to deliver the packets. Sophisticated colluding adversaries may in some cases lead to imbalance in load distribution in parts of the network. Load balancing is out of the scope of this paper, but we consider it as a future research direction.

A potential and specific attack on the AVOMIN algorithm is for an adversary to try to remove itself from the suspect list that is appended to the RREQ messages to prevent itself from being penalized by subsequent nodes. However, because the suspect list part of the RREQ message is digitally signed, the destination node $D$ will fail to verify the message, and the route going through the adversary will not be chosen.

A DoS attack is an incident in which nodes in the network are deprived of the services they would normally expect to receive. Although a DoS attack does not usually result in the theft of information or other security loss, it can cost the targeted nodes a great deal of time and money. Traditional DoS attacks, which are characterized by packet injection with the goal of resource consumptions, are out of the scope of this paper.

As previously explained, an adversary cannot apply a short route attack because of digital signature verifications conducted by the endpoints of the session. However, an adversary could apply a DoS attack that blocks correct information from reaching the endpoints by propagating low cost fabricated routes or by corrupting the suspect list. This way, a valid route might never be found because intermediate nodes only resend lower cost routes.

Handling such DoS attacks is out of the scope of this work. Nonetheless, the AVOMIN algorithm can be adapted to deal with such attacks by applying signature verification at each of the intermediate nodes (similar to ODSBR [19]).

5. COMPLEXITY ANALYSIS

In this section, we analyze the time, communication, and computation complexity of finding a misbehavior-free route using the different versions of AVOMIN. The analysis is performed over two adversarial models – a simple adversary that continuously drops packets (black hole attack) and a sophisticated adversary that knows the protocol and takes advantage of its weaknesses.

5.1. Simple adversarial model

5.1.1. Time complexity.

The analysis of the time it takes the different versions of AVOMIN to establish a route free of misbehaving nodes relies strongly on $T_{\text{det}}$, the time it takes to detect misbehaviors at the endpoints, and $T_{\text{dis}}$, the time it takes the route discovery process to find a new route. To detect misbehaviors at the endpoints, AVOMIN must monitor a total of $r \cdot W$ packets. Here $r$ is a predefined loss threshold and $W$ is the size of the sliding window. Thus,

$$T_{\text{det}} = \frac{r \cdot W}{R} \quad (5)$$

given a flow rate of $R$ packets per second [34].

The time it takes the route discovery process to find a new route depends on $T_{\text{hop}}$, the maximal time required to transmit a message by a node in the network (a single hop), and on the length (number of hops) of the resulting route. The length of the resulting route is bound by $V$, the number of nodes in the network. Consequently,

$$T_{\text{dis}} \leq T_{\text{hop}} \cdot V \quad (6)$$

In practice the length of the resulting route usually does not surpass $D$, the diameter of the network.

In its basic version, the AVOMIN algorithm iteratively detects misbehavior at the endpoints and finds alternative routes until a route free of misbehaving nodes is established. Therefore, the time $T_{\text{basic}}$ it takes the algorithm to successfully bypass the misbehaving node is as follows:

$$T_{\text{basic}} = (T_{\text{det}} + T_{\text{dis}}) \cdot I_{\text{basic}} \leq \left( \frac{r \cdot W}{R} + T_{\text{hop}} \cdot V \right) \cdot I_{\text{basic}} \quad (7)$$

where $I_{\text{basic}}$ is the number of iterations conducted by the basic version of the AVOMIN algorithm. The experimental results of $I_{\text{basic}}$ for different $\delta$ values are shown in the left bars in Figure 5.

Applying the waiting mechanism, induces an overhead to $T_{\text{dis}}$ because of the added loose time synchronization (Section 3.2). Because the maximum time synchronization error $\Delta$ accumulates in every hop, the added overhead to $T_{\text{dis}}$ is bound by $\Delta \cdot V$. Consequently, the time it takes the route discovery process to find a new route with the waiting mechanism is as follows:

$$T_{\text{dis'}} \leq (T_{\text{hop}} + \Delta) \cdot V \quad (8)$$

Figure 5. Number of iterations for basic avoidance of misbehaving nodes (AVOMIN) and AVOMIN with proactive routes calculation.
On the other hand, when using proactive route calculation (Section 3.1) the route discovery process is applied only once. Thus,

\[ T_{pro} = T_{det} \cdot I_{pro} + T_{dis} \leq \frac{\tau \cdot W \cdot I_{pro}}{R} + (T_{hop} + \Delta) \cdot V \quad (9) \]

where \( I_{pro} \) is the number of iterations conducted when using the proactive routes calculation that eliminates redundant iterations (\( I_{pro} \leq I_{basic} \)). The experimental results of \( I_{pro} \) for different \( \delta \) values are shown in the right bars in Figure 5. It is clear that the mean value of \( I_{pro} \) is very close to 1.

When multipath routing is used, all the proactively calculated routes start working at once. Thus,

\[ T_{multi path} = T_{det} + T_{dis} \leq \frac{\tau \cdot W}{R} + (T_{hop} + \Delta) \cdot V \quad (10) \]

in the case where at least one subflow free of misbehaving nodes exists among the proactively calculated routes.

Formulas 7, 9, and 10 present the time complexity of finding a misbehavior-free route by using the different versions of AVOMIN. Most of the parameters in these formulas (\( W, R, T_{hop}, \Delta, \) and \( V \)) are system parameters that are independent of the algorithm used. Thus, the given complexity analysis can be used to compare the performance of AVOMIN with other algorithms. The only nonsystem parameters are the number of iterations (\( I_{basic}, I_{pro} \)) and the loss threshold (\( \tau \)). However, Figure 5 shows that \( I_{basic} \) is relatively low (under 2 even for low \( \delta \) values) and that \( I_{pro} \) is even lower.

The value of \( \tau \) can be set by the user of the algorithm and it strongly depends on the environment in which the algorithm is deployed. For instance, bad weather may lead to a high packet loss rate even without any form of misbehavior in the network. In such cases a high threshold value must be used to prevent false misbehavior detections. Nevertheless, our analysis shows that \( \tau \) should be kept as low as the environment permits to reduce the time complexity of finding a misbehaving-node-free route.

### 5.1.2. Communication complexity.

The communication complexity refers to the number of messages transmitted in the entire network until a misbehavior-free route is established. Similarly to the time complexity, this number relies on the detection of misbehavior at the endpoints and on the route discovery process that finds a new route.

To detect misbehavior at the endpoints, AVOMIN must monitor a total of \( \tau \) (loss threshold) times \( W \) (the size of the sliding window) packets. Each packet is transmitted \( K \) times until it reaches the destination node, where \( K \) is the length of the currently used route. \( K \) is bound by \( V \), the number of nodes in the network, but in practice it rarely surpasses \( D \), the diameter of the network, for the same reasons discussed in the time complexity analysis. Thus,

\[ M_{det} \leq \tau \cdot W \cdot V \quad (11) \]

The route discovery process consists of two phases – RREQ and RREP. In AVOMIN, the RREQ messages are flooded throughout the network. When the waiting mechanism is applied, an intermediate node transmits each RREQ only once. Thus, the number of RREQ transmissions in AVOMIN is \( V \). The RREP messages in AVOMIN are backtracked via the shortest route that was chosen by the destination node. Consequently, only \( K \) transmissions of the RREP are needed. As previously mentioned, \( K \) is bounded by \( V \). Therefore,

\[ M_{dis} \leq 2V \quad (12) \]

The use of the waiting mechanism dramatically reduces the communication overhead of the route discovery process. To better illuminate the mechanism’s advantage, let us consider as an example, an algorithm that uses a similar route discovery process. The ODSBR protocol [19] does not employ a waiting mechanism, which may result in \( V(2V + 1) \) RREP messages. Moreover, in extreme cases of message delays in networks with specific topologies, the number of RREP messages in ODSBR is potentially exponential, even in the absence of an adversary [26].

The number of messages \( M_{basic} \) that are sent in the network until the basic algorithm with the waiting mechanism successfully bypasses the misbehaving nodes is as follows:

\[ M_{basic} = (M_{det} + M_{dis}) \cdot I_{basic} \leq (\tau \cdot W + 2) \cdot V \cdot I_{basic} \quad (13) \]

where \( I_{basic} \) is the number of iterations conducted by the basic version of the AVOMIN algorithm.

When using proactive route calculation, the route discovery process is applied only once. Thus,

\[ M_{pro} = M_{det} \cdot I_{pro} + M_{dis} \leq (\tau \cdot W \cdot I_{pro} + 2) \cdot V \quad (14) \]

As discussed in the time complexity analysis, \( I_{pro} \leq I_{basic} \).

When multipath routing is used, all the proactively calculated routes start working at once. Hence,

\[ M_{multi path} = M_{det} + M_{dis} \leq (\tau \cdot W + 2) \cdot V \quad (15) \]

in the case where at least one subflow free of misbehaving nodes exists among the proactively calculated routes.

Formulas 13, 14, and 15 present the time complexity of finding a misbehavior-free route by using the different versions of AVOMIN and can be used to compare the performance of AVOMIN with that of other algorithms. As in the case of the time complexity, our analysis shows that \( \tau \) should be kept as low as the environment permits to reduce the communication complexity of finding a misbehavior-free route.
5.1.3. Computation complexity.

Intermediate nodes within wireless networks (specifically in WMNs) have relatively limited computational power.Cryptographic operations, such as encryption, decryption, digital signing, and signature verification, require extensive computation. Consequently, cryptographic overhead is the most fundamental part when analyzing the computational complexity of a secure routing protocols such as AVOMIN. We do not consider the computation complexity of the endpoints, which are usually much more powerful than the intermediate mesh nodes.

In AVOMIN, no cryptographic operations are performed at all by the intermediate nodes. This is in contrast to typical security protocols. As an example, consider the ODSBR protocol [19] in which intermediate nodes have to perform signature verifications for each RREP message they receive and digitally sign it before resending. From the communication complexity analysis we know that there may be $O(V^2)$ RREP messages at each iteration of ODSBR. For each such message $O(V)$ signature verifications have to be performed for each of the preceding nodes in the RREP route, although the length of the route will usually not surpass $D$, the diameter of the network. Thus, a cryptographic overhead of $O(V^3)$ may be imposed on each route discovery process performed by ODSBR.

Moreover, while detecting misbehavior, the ODSBR protocol may enter a probing mode, which requires computationally expensive onion encryption [35]. Such extensive cryptographic operations may be unfeasible for existing deployed infrastructure of wireless mesh nodes. This means that hardware modification may have to be applied before using the ODSBR protocol.

5.2. Sophisticated adversarial model

The sophisticated adversarial model is of a node that knows the protocol and takes advantage of its weaknesses. In AVOMIN, a sophisticated adversary can cause a DoS attack that prevents the termination of the route discovery process (see Section 4).

To deal with a sophisticated adversary, signature verifications at each of the intermediate nodes have to be applied during the route discovery process in AVOMIN (similar to ODSBR [19]). Such an addition would mean that a cryptographic overhead of $O(V^2 \cdot I_{\text{basic}})$ will be added to the computation complexity of the basic version of AVOMIN or $O(V^2)$ to the proactive and multipath versions of the algorithm. Nevertheless, the cryptographic overhead of AVOMIN remains lower than ODSBR’s because of the waiting mechanism that is applied in AVOMIN and the onion encryption that is performed during the probing mode in ODSBR.

A sophisticated adversary may take advantage of the weaknesses of other algorithms too. As an example, such an adversary can cause the probing mode in ODSBR to never find the faulty link. It can do so by dynamically causing enough loss to trigger a fault, then switching to causing loss just under the acceptable threshold $\tau$ to wait out the probing mode [19]. Consequently, the protocol might keep the traffic going through the problematic path without ever initiating a new route discovery process. Moreover, the ongoing traffic between the endpoints will continuously carry probes as part of the infinite probing mode. These probes imply onion encryption that is performed by the intermediate mesh nodes. Such a cryptographic overhead on ongoing traffic in a wireless network is clearly unacceptable.

6. EXPERIMENTAL EVALUATION

The evaluation of AVOMIN is divided into two sets of experiments – the first set assesses the performance of the basic AVOMIN algorithm along with the proactive routes calculation extension, whereas the second set of experiments tests the performance of AVOMIN with multipath routing.

6.1. Experimental setup

The experimental evaluation was performed using a designated simulation tool that was written using the Java programming language. The experiments were performed on 500 simulated networks for each setting. A single session was executed in each simulated network. In all the settings the nodes were randomly placed within a area. Each node in these networks has a radio communication range of 200 m. The basic setting includes 100 nodes, but settings with higher node densities were also tested. In the basic setting a single misbehaving node was randomly selected from the nodes that are part of the shortest route between the endpoints. The distance between the endpoints (nodes $S$ and $D$) was limited to a minimum of 800 m to handle at least 5 hops in each session. In settings that test scenarios with several misbehaving nodes, the additional misbehaving nodes were randomly selected from all the nodes in the network, including the remaining nodes in the original shortest route.

6.2. Basic AVOMIN

The effectiveness of AVOMIN is demonstrated in three aspects – the failure rate of finding a misbehavior-free route, the number of iterations performed until the route is found, and the quality of that route, which is referred to as the stretch factor.

Table II displays the failure rate of finding a misbehavior-free route. We consider a failure if after 10 attempts (iterations) the algorithm does not find such a route. AVOMIN with $\delta = 0.1$ failed in 0.6% of the experiments, whereas it succeeded in all the experiments when higher $\delta$ values were used. We also display the results of the TIARA overlay routing [20] and ODSBR [19] algorithms. As previously mentioned, both these algorithms assume the network to be closed, contrary to AVOMIN.
that does not need any a priori knowledge of the network. Both TIARA overlay routing and ODSBR did not manage to find a misbehavior-free route after 10 iterations in some of the experiments. Our results also include a comparison with DSR cache, a naive algorithm that chooses alternative routes that are cached at the source node whenever misbehavior is detected at the endpoints. Such cached routes are used in DSR [21]. DSR cache fails in almost half the experiments, so it is clearly unsuitable for the task of mitigating misbehavior.

Figure 5 displays the number of iterations performed by AVOMIN until a misbehavior-free route is found for different \( \delta \) values. For each value of \( \delta \), two results are presented – the first is of the basic AVOMIN. The second shows the number of iterations performed by AVOMIN with proactive routes calculation.

The notion of iteration is different for these two versions of the algorithm. In basic AVOMIN, an iteration consists of a route discovery process and of the actual communication conducted along the discovered route until misbehavior is detected. When proactive routes calculation is applied, all the routes are known in advance and no additional route discovery is needed. Consequently, iteration in this case refers solely to the communication conducted on a proactively calculated route. As confirmed by our simulations, the use of a low \( \delta \) value leads to a better (lower) stretch factor. With \( \delta = 0.1 \) the average stretch factor is less than 1.008. Thus, the resulting routes are very close to the optimal routes. The poor results of TIARA can be explained by the practically random locations of the buddy nodes. The stretch factor of ODSBR is always 1, because ODSBR always finds the shortest available route.

A clear tradeoff between the number of iterations and the stretch factor in AVOMIN can be observed by considering both Figures 5 and 6. Consequently, if the goal is to reduce the convergence time, a high value of \( \delta \) should be chosen. On the other hand, when the main goal is the quality of the resulting route (low stretch factor) a low \( \delta \) value should be used. \( \delta = 0.3 \) seems to give a good balance when considering all the aspects.

Next, we consider the same aspects in settings with increasing node densities, from the 100 nodes of the base setting to 1000 nodes in the same 1000 \( \times \) 1000 – \( m^2 \) meter area. Figures 7 and 8 present the failure rate (after 10 iterations) and the number of iterations, respectively. In these settings we compare the performance of AVOMIN with proactive routes calculation with that of TIARA overlay

### Table II.

<table>
<thead>
<tr>
<th></th>
<th>AVOMIN ( (\delta = 0.1) )</th>
<th>AVOMIN ( (\delta \geq 0.3) )</th>
<th>TIARA</th>
<th>ODSBR</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate</td>
<td>0.6%</td>
<td>0%</td>
<td>3.2%</td>
<td>0.4%</td>
<td>42.4%</td>
</tr>
</tbody>
</table>

Definition 1. **Stretch factor refers to the ratio between the length of the resulting route and the length of the shortest existing misbehavior-free route (i.e., the optimal one).**

Figure 6 displays the stretch factor of AVOMIN by using different \( \delta \) values and of TIARA. As expected, using low \( \delta \) values results in a better (lower) stretch factor. With \( \delta = 0.1 \) the average stretch factor is less than 1.008. Thus, the resulting routes are very close to the optimal routes. The poor results of TIARA can be explained by the practically random locations of the buddy nodes. The stretch factor of ODSBR is always 1, because ODSBR always finds the shortest available route.
Avoidance of misbehaving nodes in wireless mesh networks

T. Grinshpoun, A. Meisels and E. Felstaine

Avoidance of misbehaving nodes in wireless mesh networks

Figure 8. Number of iterations under different node densities.

We only show the results for \( \delta = 0.3 \) for clarity reasons.

As shown in Figure 7, AVOMIN did not fail even once to find a misbehavior-free route in the various settings. Moreover, AVOMIN performs less iterations as the node density increases (Figure 8). The reason for this is that in denser networks there are more alternative routes of similar length between the endpoints.

TIARA also presents improved performance in denser networks, because in such networks the chance of selecting a buddy node that is disconnected from the source node or results in a route that still passes through the misbehaving node significantly reduces. Nevertheless, Figure 7 shows a quite random failure rate for TIARA in networks with 200 nodes or more. This can be explained by the inherent randomness of the buddy node choosing process. Contrary to that, the performance of ODSBR drastically deteriorates as the node density increases. The reason for this phenomenon lies in the fact that ODSBR penalizes links rather than nodes. This results in more routes passing through every node (including through misbehaving nodes). Consequently, in dense networks it takes more ODSBR iterations until a route that bypasses the misbehaving node is found. In many cases the number of iterations is larger than 10, as shown in Figure 7.

Figure 9 displays the stretch factor of AVOMIN by using different \( \delta \) values under different node densities. The stretch factor improves as the node density increases, and in networks with 1000 nodes the stretch factor is around 1 for \( \delta \leq 0.6 \). Even for high \( \delta \) values, the stretch factor is less than 1.01 in networks with at least 600 nodes. To enable a better view (and a correct scale), TIARA overlay routing was left out of the plots in Figure 9. TIARA’s average stretch factor is between 1.34 and 1.38 for the settings with 200 nodes or more.

Finally, we consider networks with several misbehaving nodes. Figure 10 displays the failure rate of finding a misbehavior-free route in networks of the basic setting with up to 10 misbehaving nodes. In TIARA overlay routing and ODSBR a failure is considered if after 10 iterations the algorithm does not find an appropriate route. For AVOMIN we consider a more restrictive notion of failure. On top of the 10 iterations threshold, we also restrict AVOMIN to only a single proactive routes calculation process. We do so because conducting an additional proactive routes calculation is a considerably more complex notion of iteration than an iteration within an existing proactive routes calculation. Nevertheless, it is still significantly less complex than an ODSBR iteration (see Section 5).

As expected, the failure rate of all the algorithms increases when the network contains more misbehaving nodes. AVOMIN has fewer failures than ODSBR in all the settings. Interestingly enough, TIARA overlay routing outperforms AVOMIN when there are at least three misbehaving nodes. The reason for this lies in the randomness of alternative routes in TIARA as opposed to the locality of the routes in AVOMIN and ODSBR. Even in cases that in the area of the original route there are several misbehaving nodes, TIARA still has a good chance of finding an alternative route by eventually selecting a buddy node that is far away from the problematic area. The downside of TIARA remains its stretch factor, as it displays similar values (for the successful attempts) to the ones shown in Figure 6 for a single misbehaving node. When considering the successful attempts of AVOMIN, the number of iterations and stretch factor remain virtually the same as the values shown in Figures 5 and 6, respectively, regardless of the number of misbehaving nodes in the network. Consequently, these results are not presented, and the interesting aspect when considering multiple misbehaving nodes remains the failure rate.
6.3. AVOMIN with multipath routing

The main advantage of using multipath routing is that it provides a robust and dynamic way to deal with alterations in the adversarial behavior. However, such robustness comes with a price of an increased communication overhead. The experiments display the imposed communication overhead with respect to different values of the DPR that is applied by the misbehaving nodes. Additionally, the dynamic distribution of packets among the subflows is demonstrated. The experiments use a sliding window of 100 packets for each subflow. The data of dropped packets that appears in the sliding windows is used for the dynamic distribution of packets among the subflows.

Commonly used for such purposes is the expected transmission count metric (ETX) [30]. ETX considers the DPR of each link in the route, which makes it an appropriate metric for either the MAC layer or for higher layer algorithms that work link-by-link. Contrary to that, the AVOMIN algorithm conducts data verifications only at the endpoints of the communication session. Thus, the DPR data of each link is unavailable on AVOMIN’s layer. We therefore turn to a similar metric that considers a route’s length and endpoints’ DPR.

Definition 2. Communication overhead refers to the ratio between the total length of routes that a packet has been sent through, including retransmissions, until it reaches its destination and the length of the shortest existing misbehavior-free route.

As an example consider the network that appears in Figure 4. Also consider that node J misbehaves by dropping packets it receives. In this case, the shortest available misbehavior-free route is SF_1 (length 5). A packet P is chosen to be routed through SF_0 (length 4), but is dropped by node J. Consequently, packet P is retransmitted, this time through SF_2 (length 6), and reaches its destination. Thus, the total length of routes that packet P has been sent through is 10. Consequently, the communication overhead of packet P is 100%.

Figure 11 presents the average communication overhead as a function of the DPR of the misbehaving nodes. The solid line represents the communication overhead of the first 100 packets (exactly the size of the sliding window). As can be seen, an increase in the DPR linearly impairs the communication overhead. The dotted brown line represents the average communication overhead after the transmission of 5000 packets between the endpoints. It is clear that after the algorithm assimilates the data of dropped packets, the effect of increased DPR disappears and the communication overhead converges to 6%. An adversary that drops about 30% of the packets maximally impairs the communication overhead. Nevertheless, the damage that such an adversary can inflict is limited to 9% communication overhead.

Figure 12 shows the convergence of the communication overhead with respect to the number of sent packets for three different DPR values. With 10% DPR, the communication overhead converges to a slightly higher value than the overhead of the first 100 packets. This negative effect is explained by the quadraticity in Formulas 3 and 4, which give more weight to the DPR values than the lengths of the subflows. Short subflows that suffer from limited DPR may have a better communication overhead than longer misbehavior-free subflows. Consequently, preferring the longer misbehavior-free subflows leads to a slightly higher overhead. Nevertheless, this preference is essential for the prompt and efficient convergence of the communication overhead under higher DPR values. The convergence with 50% DPR and with 100% DPR confirms the necessity for quadraticity in the formulas. Moreover, with 100% DPR, the communication overhead is just marginally higher than with 10% DPR.

Similar experiments to the ones shown in Figures 11 and 12 were conducted for the same networks but this time with more than a single misbehaving node. Networks in which all the subflows contained misbehaving nodes were not checked. The results of these experiments were almost exactly the same as the results shown in Figures 11 and 12.
The reason for the similarity of the results is the adaptive distribution of sent packets among the subflows. When at least one subflow is free of misbehavior, the majority of the packets are sent through this subflow.

Figure 13 demonstrates the dynamic distribution of packets on a specific example – the network shown in Figure 4. The left part of the graph presents the distribution of packets when the network contains a single misbehaving node, $J$, which drops half of the packets (50% DPR). This node resides on $SF_0$. Consequently, just 10% of the packets are routed through $SF_0$ regardless of the fact that it is the shortest of the three subflows. The center part of the graph shows how the distribution changes when node $I$ starts dropping half of the packets as well. At this point, the DPR of $SF_0$ is 75%, whereas the DPR of $SF_1$ is 50%. Although $SF_2$ is the longest of the three subflows, over 80% of the packets are routed through it. The right part of the graph demonstrates the distribution of packets in case that both nodes $I$ and $J$ stop dropping packets. The biggest share of packets is routed through the shortest subflow, $SF_0$.

7. CONCLUSIONS

A distributed algorithm that mitigates routing misbehavior is proposed. The new method re-routes a session that is carried by misbehaving nodes upon detecting routing misbehavior at the endpoints. The proposed algorithm (AVOMIN) does not expect mesh routers to sign or authenticate packets that pass through. The AVOMIN algorithm enables proactive calculation of several alternative routes. This extension along with the RREQ waiting mechanism significantly reduces the algorithm’s overhead. The proactively calculated routes can be used to perform multipath routing that provides prompt discovery of misbehavior-free routes along with survivability and continuous communicability of the network. The addition of multipath routing drastically enhances the robustness of the algorithm versus adversaries that dynamically change their behavior.

An extensive experimental evaluation compared AVOMIN with existing algorithms, ODSBR [19] and TIARA overlay routing [20]. AVOMIN needs considerably less iterations than ODSBR and TIARA to find a route free of misbehaving nodes. Once it finds such a route, it is usually close to optimal and of higher quality than that found by TIARA. Contrary to that, ODSBR always finds the shortest available route. Nevertheless, a sophisticated adversary can prevent ODSBR from ever initiating a new route discovery process, while indefinitely burdening the network with unnecessary cryptographic overhead.

Multipath routing in AVOMIN resembles the use of erasure-coding techniques, because both use multiple paths and share the similar goal of increasing the reliability and security of the network. The main difference between these approaches is in the assumptions they make. AVOMIN assumes that some group of nodes is misbehaving, whereas the other nodes relay packets properly. On the other hand, algorithms that apply coding assume that any of the paths may drop packets at any time with similar probabilities [28]. While the first assumption relates to the presence of malfunctioning or adversarial nodes, the second assumption relates to the unreliable nature of wireless links. In real-life scenarios one must deal with both phenomena, and therefore a combination of these two assumptions is called upon. This may be applicable, because the two approaches are basically orthogonal – AVOMIN chooses the paths to be taken, whereas erasure-coding manipulates the contents of the packets. However, such a combination is not straightforward because AVOMIN’s subflows are inherently not disjoint. We therefore see the merging of erasure coding and AVOMIN as an interesting direction for future work. Additionally, such combination may potentially help balance the load distribution in the network.

Another direction for future work is testing the applicability of AVOMIN in MANETs. Experiments that tested the performance of ODSBR in MANETs [19] produced encouraging results under a mobility model with various speeds. Because our experiments and analysis show the fast convergence of AVOMIN, we believe that AVOMIN will perform well in MANETs.

REFERENCES

32. Eriksson J, Krishnamurthy S, Faloutsos M. Truelink: A practical countermeasure to the wormhole...

