A Blueprint for Switching Between Secure Routing Protocols in Wireless Multihop Networks

Marc Werner, Jörg Kaiser and Matthias Hollick
Technische Universität Darmstadt
Department of Computer Science
Secure Mobile Networking Lab (SEEMOO)
Mornewegstr. 32, 64293, Darmstadt, Germany
{mwerner, jkaiser, mhollick}@seemoo.tu-darmstadt.de

Elias Weingärtner and Klaus Wehrle
RWTH Aachen University
Department of Computer Science
Chair of Communication and Distributed Systems (COMSYS)
Ahornstr. 55, 52074, Aachen, Germany
{elias.weingaertner,klaus.wehrle}@comsys.rwth-aachen.de

Abstract—A plethora of (secure) routing protocols exists for wireless multihop networks. These protocols are mostly tailored to meet the performance and security requirements of specific application scenarios. As a result, the protocols cannot easily be adapted to novel application demands, organically growing networks, etc. We argue that the modular design of routing protocols and security mechanisms can remove the key limitations of today’s monolithic routing protocols. We show the feasibility of a modular routing approach for wireless multihop networks using the example of a wireless mesh network. In particular, we demonstrate that a dynamic switch between protocol modules is possible at runtime by means of simulation as well as testbed experimentation. We further demonstrate that the security associations and the key material can be reutilized for bootstrapping novel protocol modules, thus minimizing the control overhead.

Keywords—Wireless mesh networks; Mobile ad hoc networks; Secure routing; Modular routing protocols; State transition

I. INTRODUCTION

Wireless Multihop Networks (WMNs) are currently getting deployed in the form of Mobile Ad Hoc Networks and Wireless Mesh Networks. For instance, WMNs enable connectivity in rural areas without any prior network access [1], thus replacing the need to establish tethered infrastructure to connect end users. WMNs also serve as a backbone network for large scale wireless access networks such as Google WiFi [2], as well as for a variety of application scenarios.

Currently, many configuration parameters and settings of a WMN are chosen at deployment time. This includes the choice of a routing protocol and a security architecture from a set of candidate solutions. The choice depends on the application scenario and the existing boundary conditions—current protocols for WMNs are typically tailored to a narrow scenario set—thus yielding the desired performance.

These conditions are subject to change over time. A once stable and reliable network can suddenly get unreliable due to external factors such as obstacles that obstruct wireless links or interference caused by other radio devices. Moreover, in WMNs the topology can change completely over time as mobile nodes move within the network or as the network organically grows and new applications are deployed. These changes in the environment and the network’s structure as well as its usage may require changes in the routing protocol or the security mechanisms at runtime to maintain the desired network performance. Yet, current WMN deployments do not support dynamic transitions between different routing protocols or security mechanisms.

One solution to this problem is to deploy a set of protocols and then select on demand the protocol fitting the current situation best. However, this solution imposes a significant overhead every time a switch is initiated. On the one hand, the routing information for the new protocol must be gathered and new routes established before the switch can be performed. On the other hand, the security associations between nodes have to be reestablished.

In this paper we present a solution to the problem of switching protocols during runtime and propose a blueprint of a system that can transition the state of security associations between protocols. To this end, we deviate from monolithic routing protocols and follow a modular approach to routing, thus further increasing the flexibility of our solution. Our contributions are as follows.

- We propose a system for switching between routing protocols while maintaining the established security associations.
- We implement and evaluate the proposed system in the ns-3 network simulator as well as on a 17 node WMN testbed within an office environment.

Our results show the feasibility of our approach. In particular, we demonstrate a seamless switch between secure routing protocols at runtime in a WMN in both simulation and testbed experiments.

II. CHALLENGES IN SWITCHING ROUTING PROTOCOLS IN WMNS

We need to address a number of distinct challenges to allow for dynamically exchanging a WMN routing service with another.
1) Reestablishment of Routing Information: The most fundamental issue is that changing the routing service forces certain routing information, most notably routing tables, to be newly negotiated or reestablished before the new service is operational. Such transition phases can result in service gaps at which no data can be exchanged among the nodes in the WMN.

2) Renegotiation or Mismatch of Cryptographic Identities: Routing daemons employing cryptographic identities, for instance SAODV [3], Ariadne [4] or SOLSR [5], might require keys to be renegotiated if replaced with each other. Similar to the first challenge, this might lead to service downtime as the nodes might not be able to sign or to encrypt their traffic.

3) State Transfer Between Routing Daemons: As we have already indicated before, we aim at a smoother transition between different WMN routing implementations. Our foremost goal is to reduce the service downtime by transferring established state information between the different daemons and protocols.

This exchange of state information is technically challenging for three main reasons. First, state information may not always be stored in equivalent form, resulting in the need of translating state information during the transition phase. For example, one routing daemon might operate on IP addresses while a second routing daemon might use layer-2 addresses or host names in its routing tables. Second, many routing protocol implementations are of rather monolithic nature; however, state transfers between daemons require a clear encapsulation of the system state and according interfaces: if the system state is dispersed all over a protocol implementation, it is very hard to extract and to inject the state if a transition is fired. Third, it must be ensured that the state transfer between the routing daemon implementations does not yield leakage of sensitive or private data.

4) Coordination of Switching: A core question is when one routing service should be exchanged with another one and with which one. For this purpose, we regard a monitoring service that observes the network performance using a set of pre-defined metrics as a viable solution. Still, the question remains who instructs the nodes to change the used service and when. In addition, we need to develop ways to assure that the coordination of the nodes’ transition process is protected by an adequate authentication scheme. Otherwise, potential attackers could flood the network with switching requests which would cause the entire network to fail in the worst case.

5) Node Compatibility and Service Proliferation: Finally, we need to ensure that a synchronous transition between the routing services is supported by all participating nodes. For this purpose, we require a repository that keeps track of the deployed and supported routing services at each node.

We focus on challenges 1-3 in the remainder of this paper and neglect the coordination of the switching for now.

According to the node compatibility, we assume all WMN nodes to be homogeneous, thus supporting the same set of routing protocol modules.

III. Blueprint for Switching Routing Protocols in WMNs

In this section we describe the design of our switching system. Figure 1 shows an overview of the design and the interaction of the components involved in our system.

A. MasterController

The MasterController is the central interface to control the network. When a new node joins the network it registers with the MasterController and announces the available routing protocols. The MasterController provides methods to initiate the switching process and orchestrates the switch to a new routing protocol. The controller can reside on a separate node on the network or even on a normal mesh node. The communication with the other nodes is established via a secondary channel such as a very robust but low bandwidth mesh network. A low bandwidth channel is sufficient as only status information and switching commands are exchanged but no data traffic is routed over this network.

B. Wireless Mesh Nodes

The mesh nodes form the actual network and route the data traffic from source to destination. They employ the routing protocol and security mechanism chosen by the MasterController and run the route selection and data forwarding without interaction of the MasterController.

1) OnHostController: We designed the OnHostController as a proxy between the MasterController and the protocol implementations. On one side it interfaces with the routing protocols locally available and forwards switching commands from the MasterController. On the other side it announces the available protocols to the MasterController.
We chose to introduce this proxy service to stay independent from the protocol framework used and possibly even employ different frameworks on different nodes without the need to adapt the MasterController.

2) Modular Protocols: The modular protocols handle the actual data traffic as well as manage their specific control traffic such as the neighbor announcements and the route discovery.

We encapsulate each functionality into modules at a very fine grained level and each module keeps its own state encapsulated. This helps to keep the information gathered by the module close to the origin and therefore eases the transition between different routing daemons. The transition between protocols can be seen as the transition between a set of modules where each module is either reused in the new routing scheme or is responsible for handling over its information to the corresponding new module. The state transition can be performed by storing/retrieving the information in/from a common database or alternatively using a direct exchange between the involved modules.

We strictly separate the routing and the security information to be able to transition both aspects of the protocol completely independent. This way it is possible to switch, for example, from a reactive to a proactive protocol and still keep the trust relations between nodes while the routing information might have to be reestablished from scratch.

3) Security Association Database: One of the core functions is to transition the security associations built by one protocol to another and thus avoid the need to reestablish trust relations when switching. The reuse of already existing security relations between nodes helps to conclude the transition faster and to keep the possible downtime of the network while switching to a minimum.

We introduce a Security Association Database (SAD) per node, which stores all security critical information for all protocols. It serves as a central instance on each node to provide other modules with security relevant information and each module inputs new security relations into the SAD. This avoids the need to reestablish security relations when switching protocols. No security information ever leaves the node it is stored on and every node will always keep its local information.

4) CryptoWrapper: The CryptoWrapper is a module that provides access to basic cryptographic operations such that these functions do not have to be implemented by each protocol. The CryptoWrapper acts as an interface to perform all cryptographic operations without any knowledge of the underlying libraries.

IV. IMPLEMENTATION

We developed a prototype of the switching system using the Click Modular Router [6]. In particular, Click implements the (secure) modular routing protocols, the SAD and the CryptoWrapper. The MasterController and the OnHostController were implemented as standard Unix daemons running in user space and communicating with the Click modules using sockets. This separation has the advantage that the routing framework does not have to be adapted if new features are introduced or if one of the controllers is replaced with a more advanced or distributed version. At the same time it is also possible to replace the routing protocols with another implementation without the need to replace the controllers.

As Click currently only runs single threaded and in order to keep the execution of the current configuration from blocking, handlers are not waiting for results. Hence, the handlers need to be polled for new or updated information. The polling is done by the OnHostController and once a change in status occurs the MasterController is notified.

We also implemented two secure routing daemons: the reactive SAODV [3] and the proactive SOLSR [5] protocol. For those protocols, we extended existing implementations of the non-secure versions of the protocol by adding all required security features in separate modules. Moreover, we implemented the central SAD and the CryptoWrapper within Click.

The MasterController currently only acts as the central instance to orchestrate the switching process. The decision to switch is handled manually, no automatic decision system is currently employed. For this purpose the MasterController has a command line interface where switching requests can be issued, which are then distributed to the mesh nodes in a synchronized fashion.

V. EVALUATION

We performed an experimental analysis of our system to (1) demonstrate its functionality, (2) gain insights into the limitations of our approach and (3) evaluate the performance of our system. We investigated the performance both on the control and data plane in terms of routing overhead as well as end to end (e2e) packet delay.

A. Simulator

We first studied the prototype of our system inside the ns-3 network simulator [7] using the ns-3-click bridge [8] to directly execute Click configurations from within ns-3. We set up a 802.11g ad hoc network with 17 nodes and a logical topology modeled according to our testbed topology shown in Figure 2. We adjusted the parameters of the network as well as the workload as described in Table I.

We generated the UDP data traffic using an application deployed on every node in the simulation. It also collected the data statistics by means of a callback function to calculate the end to end delay of each packet.
Table I

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi-Fi Standard</td>
<td>802.11g (54 Mbit/s)</td>
</tr>
<tr>
<td>Wi-Fi Mode</td>
<td>Ad hoc mode</td>
</tr>
<tr>
<td>Routing protocols</td>
<td>SAODV, SOLSR</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>17</td>
</tr>
<tr>
<td>Packet size, transport</td>
<td>512 bytes, UDP</td>
</tr>
<tr>
<td>Packet rate, number of flows</td>
<td>4 packets/s per flow, 4 flows</td>
</tr>
<tr>
<td>Transmission range</td>
<td>8m</td>
</tr>
</tbody>
</table>

1 Transmission range set in simulation. Range varies for testbed. In general, we matched the parameters in both simulation and testbed experiment as close as possible.

Figure 2. Node deployment in our testbed experiment.

B. Testbed

Our testbed is located in an office building and consists of 18 mesh nodes (17 of which were active in our test). The nodes are x86 compatible and feature an AMD Geode LX800 processor and an Alix 3D2 board. The location of the nodes in the building is depicted in Figure 2.

The workload and measurement traffic was generated using the tg2 [9] tool which offers a client and a server application and provides an easy way to measure the e2e packet delay by simply sending the current time stamp as the packet payload. We used the Ethernet backplane of the testbed as a secondary channel for the control messages from the MasterController to the OnHostControllers and to synchronize clocks.

C. Results

The results of both simulation and testbed experiment do not differ qualitatively, yet the testbed experiment is influenced by various limitations. Hence, we here focus on the testbed results. If not mentioned otherwise, we show data related to the flow from source node 1 to destination node 12, which typically experienced a three hop route.

Figure 3 shows the end to end delay experienced by individual data packets for a transition from OLSR to SOLSR. The first data packets are sent around \( t = 12 \) s and only experience a delay of a few milliseconds. The "hard" switch is triggered at \( t = 45s \) and it takes roughly 11 seconds before SOLSR has rebuilt its routing tables. During this period, packets are dropped, since no valid route exists. Starting from \( t = 56s \), packets are again delivered. This delay motivates the design of a seamless transition, which provides a "warm start" of the routing protocol before the switch on the data plane is performed.

It is worth noting that from \( t = 45s \) on, e2e packet delays of up to 2.5 seconds can be observed, despite the fact that the network is fair from being saturated. This delay is caused by the mesh nodes’ limited resources in terms of processing capabilities; the nodes cannot keep up with the cryptographic operations on the control plane of SOLSR. By replacing the cryptographic operations with a dummy function, we proved that the testbed can sustain the workload, showing results similar to the simulation (where computing limitations do not apply).

Figure 4 shows the control as well as data messages within the WMN for a switch between SAODV and SOLSR. At \( t = 35s \), SOLSR starts establishing routing tables. The switch between both protocols is performed at \( t = 45 \). OLSR HELLO messages omitted for clarity.

Figure 3. End to end delay of data packets in seconds for a "hard" switch between OLSR and SOLSR. The switch is triggered at \( t = 45s \). Red (light gray) crosses indicate dropped packets.

Figure 4. Number of control and data messages per 500ms intervals for a "soft" switch between SAODV and SOLSR. At \( t = 35s \), SOLSR starts establishing routing tables. The switch between both protocols is performed at \( t = 45 \). OLSR HELLO messages omitted for clarity.
leads to an increased jitter in the e2e delay of the data packets. The network enters the second phase at $t = 45$, where it switches all data traffic from SAODV to SOLSR. We observe a more bursty performance of SOLSR compared with SAODV, which can be explained by the higher control message overhead (signature and topology control messages) that is competing with the data traffic and with the mesh nodes operating close to their processing capacity due to the necessary cryptographic operations.

VI. DISCUSSION

We have presented selected results that show the feasibility of our approach to switch between WMN routing protocols at runtime, reusing the existing security associations.

The results we obtained from the simulation and testbed experiments are qualitatively similar. Yet, we observed a number of interesting phenomena during the testbed experiments. We discuss these aspects in the following.

For the given secure routing protocols, the asymmetric cryptography resulted in very high processing loads on our testbed nodes. Further security operations induced by a protocol switch such as key renegotiation put additional burden on network and nodes, thus supporting our work to reuse security associations. We further conclude that the deployment of secure protocols can effectively lead to a denial of service in the network if nodes do not adhere to certain rate limits for issuing/forwarding control messages. This can be remedied by either rate limiting of control messages, more powerful nodes, lightweight cryptography or dedicated cryptographic co-processors.

VII. BACKGROUND AND RELATED WORK

We present related work in the areas of modular routing, routing frameworks and secure routing protocols. We focus on protocols that have been implemented and deployed in testbeds or real networks.

A. Modular Routing Frameworks

*marnetd* [10] is a framework to dynamically load and unload routing protocol modules from the protocol stack. The selection of the used protocol is done by the application currently running and can therefore specifically be adopted to the communication needs.

*Manetkit* [11] is dedicated to create routing protocols for MANETs and brings a plugable protocol stack together with run-time deployable components. It offers a high flexibility in defining and combining different routing algorithms. Unfortunately, at the time of this work the implementation of the Manetkit framework was not publicly available.

The *Click Modular Router* [6] is a well established but still evolving framework to create software routers. It provides functionality from the link layer up to the application layer and can be easily extended by creating customized elements. A router’s configuration is created as a directed graph which is traversed by packets. The flexibility to control the packet flow on different protocol layers and the extensible API make the Click Modular Router a framework for implementing a wide variety of protocols.

However the modularity of security functions in WMNs has not been studied yet.

B. Secure Routing Protocols

In Table II we give an overview of selected secure routing protocols and their security associations. Within this paper, we focus on demonstrating the feasibility of reusing security associations without requiring to transform state information, which might be necessary if switching between arbitrary protocols.

Hence we have chosen to implement secure versions of the *Optimized Link State Routing (OLSR)* [15] and the *Ad-hoc On-demand Distance Vector (AODV)* [16] protocol for our prototype. These protocols share common security associations and the base protocols are available for the Click Modular Router.

OLSR is a proactive routing protocol designed for large and dense mesh networks. It uses multipoint relays to forward broadcast messages carrying routing state information. OLSR itself was not designed with security in mind, but there are several extensions such as Secure OLSR [17] and the work by Clausen et al. [5], which uses public key cryptography to secure the control traffic. We chose the security model by Clausen et al. as it is compatible with the security model of SAODV described below.

*AODV* on the other hand is a reactive protocol which only establishes a route between two nodes in the network on demand, i.e., if there is actual data traffic to be forwarded between these two nodes. AODV was not designed with security in mind either, but a security extension, SAODV, has been proposed [3]. It utilizes public key cryptography to secure the route discovery messages.

To the best of our knowledge there exists no previous work that investigates practical (modular) switching between (secure) routing protocols in WMNs.

---

**Table II**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>sym.s–d</th>
<th>sym.nbr</th>
<th>pub.</th>
<th>chain</th>
<th>sym.all</th>
<th>sym.abr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ariadne [4]</td>
<td>X</td>
<td>(X)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAODV [3]</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEAD [12]</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASTOR [13]</td>
<td>X</td>
<td>(X)</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP [14]</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOLSR [5]</td>
<td>(X)</td>
<td>(X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 Symmetric key shared between source and destination node.
2 Public key of all nodes in the WMN.
3 Seed hash chain element for hash chains of all nodes in the WMN.
4 Symmetric key shared between all nodes in the WMN.
5 Symmetric key shared between source and destination node.
6 Parenthesis mark optional keys or a dependence on the implementation.
VIII. CONCLUSION

In this work we presented a system allowing to switch between secure routing protocols. After having discussed the key challenges in switching protocols at runtime, we have defined a blueprint of the switching system. It enables the seamless transition between protocols at runtime by leveraging state information such as the nodes’ security associations. To this end, we have introduced controller components (MasterController and OnHostController) as well as a joint Security Association Database. We have implemented a prototype based on the Click Modular Router framework, which enables a switch between the secure WMN routing protocols SAODV and SOLSR. While our protocol implementations are modular, thus allowing for switches on sub-protocol level, in this work we have focused on transitions between protocols as a whole. Our proof-of-concept implementation has been tested using both simulation as well as testbed experimentation and the results demonstrate the feasibility of our approach.

IX. FUTURE WORK

Sharing and reusing state information—such as protocol state or security associations—is key to allow for seamless transitions. As future work we plan to investigate more complex transitions and transitions with finer granularity. We will put particular emphasis on how to securely yet efficiently transform and adapt state information that cannot be directly reused, one example being cryptographic key material, which might not fit the security architecture in question and which should not be directly reused with a novel cryptographic algorithm.

In our experiments we simplified the control of the transition to be centralized and manually executed. We plan to investigate decentralized schemes in future work. Moreover, in practical deployments, a suitable decision metric which signifies whether to switch or not and which transitions to perform, needs to be identified. The stability and robustness of the transitions require further investigation.

ACKNOWLEDGMENT

This work has been co-funded by the DFG as part of the CRC 1053 MAKI (www.maki.tu-darmstadt.de), by LOEWE CASED (www.cased.de), and by BMBF EC SPRIDE (www.ec-spride.de).

REFERENCES
