SECURING THE SCALABLE SOURCE ROUTING PROTOCOL

Kendy Kutzner, Christian Wallenta and Thomas Fuhrmann

System Architecture Group, Universität Karlsruhe (TH)
76128 Karlsruhe, Germany
Tel. +49 721 608 6731 Fax: +49 721 608 7664
eMail: {kutzner|wallenta|fuhrmann}@ira.uka.de

Abstract

The Scalable Source Routing (SSR) protocol combines overlay-like routing in a virtual network structure with source routing in the physical network to a single cross-layer architecture. Thereby, it can provide indirect routing in networks that lack a well-crafted structure. SSR is well suited for mobile ad hoc networks, sensor-actuator networks, and especially for mesh networks. Moreover, SSR directly provides the routing semantics of a structured routing overlay, making it an efficient basis for the scalable implementation of fully decentralized applications. In this paper we analyze SSR with regard to security: We show where SSR is prone to attacks, and we describe protocol modifications that make SSR robust in the presence of malicious nodes. The core idea is to introduce cryptographic certificates that allow nodes to discover forged protocol messages. We evaluate our proposed modifications by means of simulations, and thus demonstrate that they are both effective and efficient.

Keywords: Routing, Security, Cross Layer, Communication Abstraction, Mesh Networks

1 Introduction

The world of telecommunication has changed greatly over the last decades. The Internet has replaced circuit switching by packet switching and thereby substantially lowered equipment and deployment cost. Meanwhile, well structured public infrastructures are more and more complemented by unstructured local infrastructures, be it multi-homed global company networks or grass root wireless LAN clouds. *Mesh networking* is one of the terms that describe such new networking developments. The term describes the combination of mobile ad-hoc networks, different wireless infrastructures and wired infrastructures to provide seamless connectivity to the end-user.

Routing in such scenarios is inherently difficult. This is even more the case in so-called sensor-actuator networks where very low resource nodes are randomly deployed to monitor physical parameters and react accordingly. In [8] we proposed a memory and message efficient routing protocol for large unstructured networks, the *scalable source routing* (SSR) protocol. Unlike routing algorithms that are based on the Dijkstra or Bellman-Ford algorithm, SSR does not need large routing tables. It is therefore immune against problems of routing table explosion. Unlike many ad-hoc routing algorithms, SSR does not need to flood the network with control messages. It is therefore control message efficient.

Originally, SSR aimed at sensor networks where resource efficiency was the foremost goal. Security was not considered an issue since the network was supposed to consist of trusted nodes only. But if SSR is to be employed in mesh networks, too, the protocol must also be secure with respect to malicious nodes.

In this paper we present an analysis of the possible attacks against the routing integrity of networks using the SSR protocol and propose protocol modifications that harden the protocol against these attacks. Simulation results evaluate the performance of the so modified protocol.

2 Related Work

The idea to combine MANET routing with structured P2P protocols has been independently proposed by different groups in recent years. CrossROAD [2] hashes the IP addresses of participating nodes, stores them in a routing table, and uses this table to create the overlay topology. Since CrossROAD is based on OLSR [12] it inherits the same scalability limitations. In contrast, it has been shown in [9] that SSR can scale well beyond 100,000 nodes.

MADPastry [21] combines Pastry [18] and AODV [15]. It uses so called random landmarks to assign virtual addresses to nodes. Since these virtual addresses are location-dependent, they change frequently in presence of node mobility. This creates significant overhead. Moreover, the virtual addresses serve only as guidance and the protocol has to revert to flooding if the destination address is not found. Since SSR uses location-independent addresses it reduces the control message overhead. Moreover, SSR avoids flooding during the routing process. Thereby, SSR can further reduce the traffic load in the network.

Ekta [16] is an enhancement of DPSR [11]. DPSR is a cross-layer combination between Pastry and DSR [13]. Contrary to SSR, both require flooding to discover routes.

SSR [9] is based on ideas that were first presented in references [14] and [4]. There, source routing and principles from overlay networks are combined in a scalable manner. A subsequent analysis [5] proved that this approach always converges to a globally consistent network. Later on, SSR was implemented in an embedded wireless LAN router [6] to show that these ideas can be actually implemented in real world systems.

Security aspects in overlay networks have been studied extensively. See [3] for an overview that especially describes the security and stability challenges in overlay networks with malicious nodes. In such networks, it is especially important to use secure node identifications that can withstand attacks like the Sybil attack [7]. Such secure node identifications are essential for the work in this paper, too.

In [10] Hof et.al. present a security architecture aimed at sensor networks. This work is based on Content Addressable Networks (CAN, [17]) and contains some valuable insights on the use of cryptography on resource limited devices.

A general approach to secure communication in ad-hoc networks was presented in [1]. Balfanz et.al. provide basic authentication based on public key cryptography as well as inexpensive alternatives derived from the Guy Fawkes protocol. In our work, we rely on public key cryptography, because we don't assume the location limited side channel as in [1].

3 Scalable Source Routing

SSR is a general routing protocol that combines ideas from peer-to-peer overlay routing with source routing. Unlike typical peer-to-peer overlays, SSR operates in the network layer. SSR is based on two key ingredients: 1. Each node maintains a cache that stores source routes on a least recently used (LRU) basis. 2. When a node needs to route packets to a destination for which the cache does not yet have a source route, such a route is constructed iteratively (see below).

It has been demonstrated [9] that the cache can be very small. A cache storing source routes that are composed of up to 255 unique nodes suffices for many network topologies. Since the cache can be organized as a tree, a node is able to produce a source route to any of the nodes in its cache.

SSR assumes all nodes of the network to be identified by unique identifiers, for example, MAC-48 addresses. Conceptually these identifiers are thought to form a virtual ring, i. e. the ID space is treated as if it was circularly glued together. Obviously, the ring will be only sparsely populated. Within this ring, we can define a metric to measure virtual distances, e. g. the numerical difference of the respective IDs. This metric induces the concept of virtual neighborhood, i. e. two nodes are said to be virtual neighbors if no other node has an ID between the two respective nodes' IDs. It is important to note that this virtual identifier ring is independent of the actual network graph, i. e. no assumptions about a correlation of the nodes' IDs and their physical location or their connections are made.

Besides the metric in the virtual ring, SSR employs a second metric measuring physical distance in the network graph. Typically this will be the length of a source route between two nodes. SSR uses both metrics for its routing decisions.

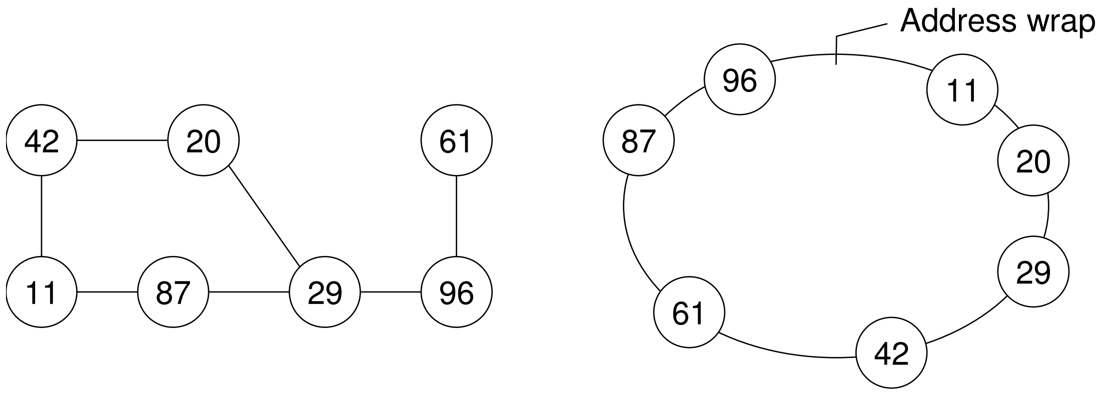

Figure 1: Example of a physical network topology and the according virtual ring

Upon bootstrapping, nodes discover their physical neighborhood by broadcasting `Hello` messages, or, if available, by some link layer mechanisms. The corresponding trivial one-hop source routes are entered into the cache. From that, the nodes then construct the non-trivial source routes to their virtual neighbors. This is done as follows:

Each node selects its presumed virtual neighbors from its cache and sends them a `NeighborNotification` message. Upon reception of such a message, the receiving node either agrees with the sender to be its virtual neighbor, or it knows some better virtual neighbor. Let's call the sender *A*, the receiver *B*, and the better neighbor *C*. By construction the received message contains a source route from *A* to *B*. Also by construction, *B* caches a source route to *C*, since otherwise it would not know of *C* at all. From both, *B* can construct the source route from *A* to *C* and send an according `NeighborUpdate` message to *A*. (See [9] for details.)

It has been shown that this mechanism brings the network into a consistent state. Typically, this state will be globally consistent, so that each node caches source routes to its virtual neighbors. In rare cases an additional mechanism is required [4] to guarantee consistency. Moreover, in all studied cases consistency is reached quickly with the exchange of only few `NeighborNotification` and `NeighborUpdate` messages for each node.

Once each node caches source routes to their virtual neighbors, any node can reach any destination by following the SSR routing rules described below. Before global consistency is reached, routing success cannot be guaranteed. Depending on the degree of consistency, a some packets might be deliverable anyway.

If a node has previously communicated with a given destination, it can directly produce a source route from its cache. While forwarding the packet, this source route is pursued in the obvious way. When a node cannot retrieve the entire source route to the destination from its own cache, it constructs a source route to some other, so-called *intermediate*, node. The message is then forwarded to that intermediate node, which in turn takes care of the message and appends another source route to the message. It can be shown that this process converges, thereby producing the full source route from the original source to the final destination.

The selection of such intermediate nodes is governed by a three step process: 1. The next intermediate node must be virtually closer to the final destination than the current node. 2. Of all nodes satisfying the first condition, those nodes are chosen that are physically closest to the current node. 3. From these, the one is chosen that is virtually closest to the final destination. In rare cases this rule does not produce one but two nodes with identical virtual and physical distance from the current node. Both can be chosen equally well. — In other words, a node I_n selects a next intermediate node I_{n+1} so that $d_v(I_{n+1},D) < d_v(I_n,D)$, $d_p(I_n,I_{n+1}) = \min$, and $d_v(I_{n+1},D) = \min$ where D is the destination. The three conditions apply in strict order.

Before we analyse SSR's security issues, we briefly illustrate its workings with an example. Assume that the network depicted in fig 1 is in such a state that each node happens to store only its physical neighbors and the source routes to its virtual neighbors in its route cache. This is atypical, since the nodes' caches usually contain much more information, but it simplifies the example.

Assume that node 20 wants to send a message to node 61. As defined in this example, it holds source routes to nodes 11 and 29 (virtual neighbors) and knows that 29 and 42 are its physical neighbors. Applying the routing rules stated above, node 20 chooses to forward the message to node 42 since it is physically closest to itself and virtually closest to node 61. Node 42 has by definition a source route to 61 (virtual neighbor) and can thus append the rest of the source route to the message. Assume that 42's route to 61 happens to be 42-20-29-96-61. Then, appending it to 20-42 would lead to a loop. This loop can be easily detected and cut out of the resulting source route. As a consequence, 61 receives a message containing the route 20-29-96-61, which happens to be the shortest path route. (Shortest paths are not always found. See [9] for details.)

4 Possible Attacks

Since the original SSR protocol trusts in all the information contained in the received control messages, a malicious node could arbitrarily modify node addresses and routing information. This leads to three major types of attacks on a routing protocol like SSR:

- A rogue node can manipulate other nodes so as to route traffic around it, thereby saving its own resources, e. g. energy.
- Conversely, it may attract traffic and use this traffic to perform further attacks like sniffing, modifying or deleting this traffic. For example, a malicious node can issue `Hello` messages with many different IDs.
- It can disturb the overall performance of the routing system (denial of service attack).

Figure 3 shows the effects of such an attack on the network. Here, the attacking node forges addresses and modifies its own routing rules with the goal to disturb the network and attract a lot of traffic to itself. It can be seen from the figure that the overall ratio of correct successors is dropping rapidly while the number of unsuccessfully delivered messages increases steadily.

5 Securing the SSR protocol

The improvement proposed by this paper is based on two key ideas [20]:

1. First, the nodes are equipped with a public/private key pair. Then the nodes' IDs are cryptographically secured by a central authority. Such an authority can be the owner of a node, the manufacturer or any other trusted entity that can be held responsible for duly interworking with the network. The authority's signature binds a node's public key to its ID. The resulting ID certificate has to be installed in the node. As a result, trust moves from the owner, manufacturer, or operator to the respective devices. Note that this operation is only necessary once, namely for the introduction (production, purchase) of new nodes. During normal operation, this central entity is not required.
2. The second key observation is that it is necessary to trust in the links of the network. This is achieved by the following three measures:
   - `Hello` messages need to be acknowledged by the respective peers with a `HelloReply` message. Further, both the `Hello` message as well as the acknowledgment contain signatures on the loosely synchronized time-stamps. Only thus, trust between physically neighboring nodes can be established and the two peers can be assured that an alleged physical link really exists.
   - Both peers of the physical link then sign the so established fact and thereby provide the basis for other nodes to trust in the fact that this link exists. Such a signature forms, again together with a time-stamp, a one-way certificate for this link. The one-way certificate is sent to the other end of the link with a `SendLinkCertificate` message. Both certificates together yield a link certificate.
   - These link certificates can be transported through the network to prove the authenticity of the link to any other node.

Third parties may need the ID-certificates in order to check the validity of link certificates. Since ID-certificates have a rather long lifetime, they are not attached to every link certificate but transfered on demand. A demanding node may send a `GetIdCertificate` message to request one or more ID certificates. Because of their long lifetime, ID-certificates can be cached easily. Any node that has some of the requested certificates in its cache will answer with appropriate `SendIdCertificate` message.

With the ability to certify links, any route can be certified too, by concatenation of the certified links. Figure 2 shows the concept. There, step 1 shows the certification of node IDs by the certification authority (CA). Step 2 is

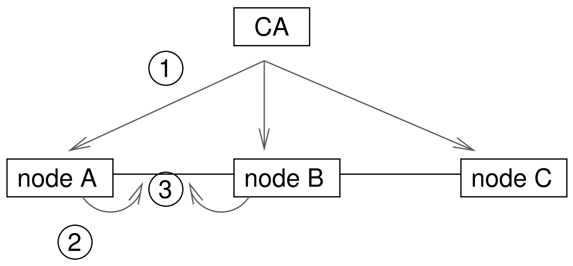

Figure 2: Usage of different certificates

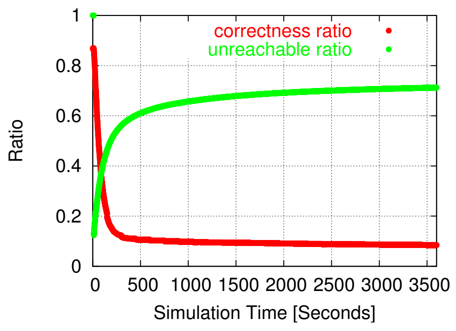

Figure 3: Performance (unsecured protocol)

the generation of one-way certificates. Two one-way certificates are joined to form a link certificate in step 4. The `NeighborNotification` and `NeighborUpdate` messages described in section 3 are now equipped with these verifiable routes. Furthermore, they are signed and time-stamped to show they really originate at the given source and to prevent replay attacks. To ensure the proper reception of `NeighborNotification` messages, these messages are now acknowledged with a signed `NeighborNotificationAck` message. A node does not change its source routes to virtual neighbors until it receives a proper `NeighborNotificationAck` message. If the acknowledgment is lost, the notification process is repeated.

With these improvements, rogue nodes are no longer able to (a) pose as other nodes and (b) introduce false routes into the system. As a consequence, it is ensured that all nodes have genuine routes in their caches and also forward only genuine routes.

Note also that all routing information is assumed to be soft-state, i. e. there is no explicit revocation of certificates. Accordingly, all certificates carry a time stamp. When the time stamp reaches its lifetime (a decision made autonomously on every node), the certificates are discarded. Note that this requires only loosely synchronized clocks on all nodes. To ensure that the necessary certificates are available during packet forwarding, the exchange of `Hello` /`HelloReply` and `NeighborNotification` /`NeighborNotificationAck` is repeated well before the certificates expire.

6 Evaluation

To evaluate our concept, the secured as well as the unsecured variant of SSR was implemented as Omnet++ [19]. Since computation time is a limited resource and the signature generation and verification are CPU-intensive processes, CPU usage has to be considered in the simulation. We modeled CPU usage as an additional delay for all further messages.

As it can be seen in figure 4, the new secured SSR protocol can withstand the attacks described earlier. The only significant difference a malicious node can make is a small increase in message delivery delay. The correctness of successor pointers reaches the desired 100% value and almost all messages are delivered correctly.

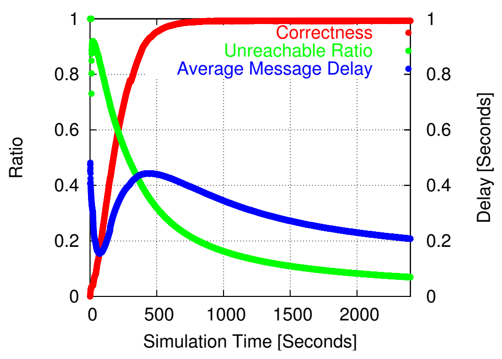

Figure 4: Performance of the secured protocol

7 Acknowledgments

We would like to thank German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for support of this work under grant number FU/448.

8 Conclusion

In this paper we described the security extensions to the Scalable Source Routing protocol. SSR itself is a routing and forwarding protocol working on top of the link layer, primarily targeting networks without predefined structure like mobile ad-hoc or mesh networks. It is based on ideas from structured overlay routing and works with sources routes. However, the basic protocol is prone to a number of easy to mount attacks. With the security extensions presented in this paper, SSR can be secured to a great extent. Especially, it can be ensured that only correct routes are established. The evaluation showed that the tried attacks are successfully prevented and give an indication of how much overhead the security mechanisms require.

References

[1] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks. In *Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California*, Feb. 2002.

[2] E. Borgia, M. Conti, F. Delmastro, and E. Gregori. Experimental Comparison of Routing and Middleware Solutions for Mobile Ad Hoc Networks: Legacy vs Cross-Layer Approach. In *Proceedings of the ACM SIGCOMM '05 Workshops*, pages 82–87, Philadelphia, PA, USA, Aug. 2005.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*, pages 299–314, New York, NY, USA, 2002. ACM Press.

[4] C. Cramer and T. Fuhrmann. ISPRP - A Message-Efficient Protocol for Initializing Structured P2P Networks. In *Proceedings of the IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks (IEEE IWSEEASN'05)*, Phoenix, AZ, April 7-9 2005.

[5] C. Cramer and T. Fuhrmann. Self-Stabilizing Ring Networks on Connected Graphs. Technical Report 2005-05, University of Karlsruhe (TH), Fakultaet fuer Informatik, Mar. 2005.

[6] P. Di, M. Marcon, and T. Fuhrmann. Linyphi: An IPv6-Compatible Implementation of SSR. 2006. To appear in: Proceedings of the Third International Workshop on Hot Topics in Peer-to-Peer Systems, Rhodes Island, Greece.

[7] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer-Verlag, 2002.

[8] T. Fuhrmann. Scalable routing for networked sensors and actuators. In *Proceedings of the 2nd Annual IEEE Comm. Soc. Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, California, Sept. 2005.

[9] T. Fuhrmann. Scalable routing for networked sensors and actuators. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Sept. 2005.

[10] H.-J. Hof, E.-O. Blaß, and M. Zitterbart. Secure Overlay for Service Centric Wireless Sensor Networks. First European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), Aug. 2004.

[11] Y. C. Hu, S. M. Das, and H. Pucha. Exploiting the synergy between peer-to-peer and mobile ad hoc networks. In *Proceedings of HotOS-IX: Ninth Workshop on Hot Topics in Operating Systems*, Lihue, Kauai, Hawaii, May 2003.

[12] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Proceedings of the 2001 IEEE International Multi Topic Conference (IEEE INMIC)*, pages 62–68, Lahore, Pakistan, Dec. 2001.

[13] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, 353:153–181, Feb. 1996.

[14] K. Kutzner, C. Cramer, and T. Fuhrmann. Towards Autonomic Networking using Overlay Routing Techniques. In *Proceedings of the 18th International Conference on Architecture of Computing Systems (ARCS '05) - System Aspects in Organic and Pervasive Computing*, Innsbruck, Austria, March 2005.

[15] C. E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, USA, Feb. 1999.

[16] H. Pucha, S. M. Das, and Y. C. Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, English Lake District, UK, Dec. 2004.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the SIGCOMM 2001 conference*, pages 161–172. ACM Press, 2001.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware) 2001*, Heidelberg, Germany, Nov. 2001.

[19] A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic*, 2001.

[20] C. Wallenta. Design and evaluation of a secure self-organizing routing protocol. Student thesis, System Architecture Group, University of Karlsruhe, Germany, Aug. 2005.

[21] T. Zahn and J. Schiller. MADPastry: A DHT Substrate for Practicably Sized MANETs. In *5th Workshop on Applications and Services in Wireless Networks (ASWN 2005)*, Paris, France, June 2005.