

# An Anonymous Multicast Routing Protocol For Mobile Ad Hoc Networks

Somayeh Taheri, Dieter Hogrefe  
Institute for Informatics-Göttingen, Germany  
{taheri, hogrefe}@informatik.uni-goettingen.de

## ABSTRACT

Anonymity is still a challenging issue in mobile ad hoc networks. Specially although operating as groups is required by many of ad hoc applications, there is only a little work on anonymous multicast routing algorithms. In this work we propose an *Anonymous Multicast* routing protocol for ad hoc networks called AnoMul based on the underlying unicast routing protocol ANODR proposed by J. Kong et al. AnoMul is a mesh based multicast routing protocol in which we propose some mechanisms to achieve our privacy goals.

## 1. INTRODUCTION

Mobile ad hoc networks (MANETs) are infrastructure-less networks used in areas where rapid network configuration is needed, such as battle field communication. The lack of a trusted centralized authority, limited resources and the broadcast nature of wireless links make these networks susceptible to security threats. Specially in hostile environments it is important to support the communication with anonymity, i.e. hiding the relationships and identities of the nodes from the adversary. On the other hand the notion of collaborating teams, such as sharing information or showing audio/video to the members causes the high importance of *multicast* in MANETs. In ad hoc networks efficient anonymity is still an elusive issue. Furthermore, in multicast communication achieving privacy is more complicated, e.g. the sender should hide not only from one receiver but from a group of receivers. But there is a little work on anonymity multicast even for fixed networks. In this proposal we propose briefly the idea of an anonymous multicast routing protocol for MANETs, called AnoMul. The idea of identity free communication proposed in [1] is exploited for multicast scenario. We apply our privacy ideas such as hiding the type of some messages to improve the privacy.

## 2. ANODR REVIEW

ANODR, [1], is a unicast anonymous MANET routing protocol. ANODR is identity free, i.e. it does not use the nodes' identities but it exploits a route pseudonymity approach to

address the *route untraceability* problem. The source node initiates a RREQ packet containing an anonymous global trapdoor and an onion. The onion is constructed by the sender encrypting some kernel with its secret key. Each intermediate node will add its self-aware layer to the onion. The global trapdoor is encrypted by the destination's public key, decryptable only by the destination. When a node receives the RREQ message it tries to open the trapdoor using its private key to check if it is the intended destination. Otherwise it generates a public/private key pair and replaces its one time public key in the appropriate field in RREQ message and broadcasts the packet to its neighbors. The next node performs the same modification and records the one time public key of the previous node to use it in RREP phase.

**RREQ:**  $\langle RREQ, seq\#, global\ trap, onion, PK - 1time \rangle$   
When the destination receives the RREQ message it will initiate the RREP message which includes a random route pseudonym encrypted by the one time public key of the previous node and the proof of the global trapdoor opening and the onion encrypted by the route pseudonym as a secret key. Every node on the route uses its one time public key to extract the upstream node's route pseudonym. Then the node uses the route pseudonym to extract the onion. It decrypts its own layer from the onion and records the correspondence between its own route pseudonym and its upstream node's one and then forwards the packet. When the source node receives the RREP packet and reveals the onion core it sent out a while ago, the anonymous virtual circuit establishment (storing the route pseudonyms) is done. The route pseudonyms will be used as secret keys between every two consecutive nodes en route in data forwarding phase.

**RREP:**  $\langle RREP, \{K_{seed}\}_{PK-1time}, f_{K_{seed}}(Proof_{des}, onion) \rangle$

## 3. ANOMUL MESH CONSTRUCTION

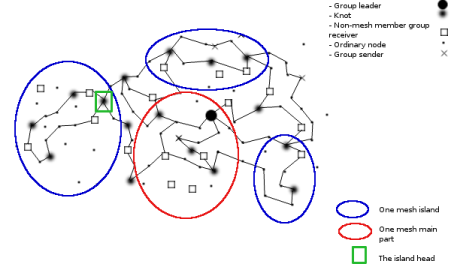
AnoMul is a mesh based multicast routing protocol. The source nodes and the group receivers join the mesh based on the underlying unicast ANODR protocol by initiating JREQ (join request) and waiting to receive a JREP (join reply) message from other mesh members. JREQ and JREP packets are of the same format of route discovery messages in ANODR, but the multicast trapdoor is encrypted with a group key,  $K_g$  which is the group key shared among the group members (with a suitable re-keying mechanism). Also we use our not already published message type unification mechanism called RDIS (Route DIScovery) to unify the REQ and REP messages format, in order to prevent the adversary from distinguishing between them. Otherwise the adversary

will be able to distinguish and trace REQ and REP messages and then track their source location. He will consider the source of the REQ messages as group members trying to join the mesh and the source of the REP messages as mesh members. In RDIS the REQ and REP messages format are changed in such a way that only the nodes en route can use their keys to distinguish their types and any other node including the adversaries can not. RDIS is briefly introduced in the appendix A . A similar message type unification called PType is also proposed in this protocol for the other message types for the sake of location privacy as well. If a joining node receives any JREP messages it realizes that a multi-cast mesh already exists for that group. Otherwise the node considers itself as the *leader* of the mesh. Next group receiver would find its route to the leader to join the mesh. We call a group receiver which is already joint to the mesh as a *knot*. The next joining receivers should join the group finding their routes to the existing *knots*. If the joining node receives more than one JREP messages, it will join the mesh through the first two among those (not just the first one), so the protocol will have more robustness. This is done by sending a *Route Confirmation message* (RCM) over the first two routes by the joining node to indicate to the nodes on those routes to preserve the recorded pseudonyms.

**RCM:**  $\langle Ptype, N_i, f_{K_{seed}}\{all, i, confirmation, K_{seed}\} \rangle$   $N_i$  is the route pseudonym generated by applying a one-way function  $i$  times on  $K_{seed}$  ( $i$  is the number of packets transmitted over the hop). The next node looks up for the pseudonym in its routing table. If found, the node decrypts RCM using the correspondent  $K_{seed}$ . The extracted message tells the node to maintain the recorded pseudonyms. The tag *all* indicates that this message can be seen not just by the group members but by every node who sees this tag, without need to the group key. The nodes en route modify the pseudonym in the message to the next hop's pseudonym and forward this message. Every *knot* maintains a counter and increments that after sending or receiving any confirmation message. So this counter indicates the number of connections to the mesh that the *knot* has, which will be used later. Now data packets can be spread over these routes. *Knots* forward them over the anonymous routes between themselves till it is delivered to every *knot*. Data forwarding between each two *knots* is similar to unicast ANODR, but the data payload is also encrypted with  $K_g$ .

**Multiple Forwarding Issue:** As explained before, every *knot* confirms just two route pseudonyms, let's say  $N_i$  and  $N'_i$ , on the first hop of its first two discovered routes. A *knot* also might get more connections later, when other *knots* join the mesh through that. Therefore any *knot* may have several shared pseudonyms and therefore needs to forward any received data packet several times (with different route pseudonyms). To avoid this problem we propose the *united route pseudonym* approach in which each *knot* changes all of its pseudonyms shared with other *knots* to its first received route pseudonym  $N_i$  considered as the united pseudonym. When the *knot* confirms the route pseudonyms of the first two discovered routes then it should change the first hop's route pseudonym of the second one,  $N'_i$ , to the first one,  $N_i$ . Later on, when this *knot* is the connection point of newly joining *knots* to the mesh, it will send the *united route pseudonym*,  $N_i$ , in JREP as the shared route pseudonym. The mesh *leader* maintains the mesh by sending a periodic message to the *knots* over the routes. If a *knot* does not

receive these periodic messages longer than some threshold it realizes that it has lost all of its connections to the main part of the mesh and it needs to join it again. We refer to any part of the mesh which includes the leader as a main part, and any part of the mesh which does not include the leader is referred to as a mesh island (Figure 1). If an island is disconnected from the mesh because one *knot* loses its direct connections to the main part of the mesh (called *island head*) then we use some efficient mechanism to reconnect the island. The idea is to find a new route to the mesh just by the island's *head*, not by every *knot* in the island.



**Figure 1:** Examples of mesh islands and main parts

Briefly, in this mechanism the disconnected *knots* initiate a *connection-check message* over their routes to ask the *knots* next to them if they are still connected to them. The number of replies they receive shows the number of connections to the mesh they still have. The *knot* compares the number of replies to the value of its connection counter explained before and if it is less than that the *knot* considers itself as an *island head* and it will join the mesh again.

### 3.1 Privacy Mechanisms

Since the underlying unicast routing protocol is identification free, so the privacy issue of AnoMul is location privacy. Shortly, the adversary model here is a powerful adversary with unbounded eavesdropping capability but bounded computing and node intrusion capability. To support location privacy, besides using message type unification mechanisms (which prevents the adversary from distinguishing between different message types and tracing the location of nodes who are expected to initiate that message types) we also propose some other privacy mechanisms. For group sender privacy we propose forming a group of *semi-senders* for each real sender. Briefly the group of *semi-senders* is formed by a number of knots whose behavior is just like the real sender. The real sender chooses them among its neighbor knots by sending a special message to them (we ignore the details here because of the limited pages). Then each data packet will be sent to the mesh randomly by one of the semi senders or the real sender. Therefore the anonymity set of each sender increases from one to number of semi senders plus one.

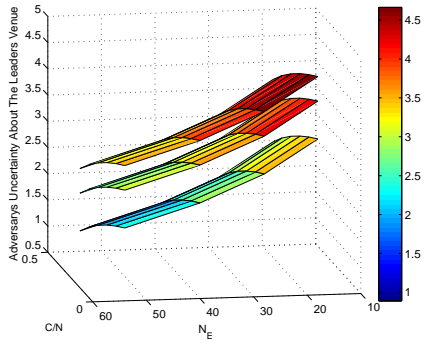
**Some Anonymity analysis:** The colluding adversarial nodes can eavesdrop to detect periodic *Ptype* messages and infer that they should be the periodic leader maintenance messages. They can measure the different latencies of such messages at different points in the network field at each leader period. Different delays indicate the different number of hops from the leader. So in this attack the adversary will consider the nodes who forward such a message earlier than the others (in each period) to be closer to the leader and use this information to find the leader's location. In this attack scenario the adversary divides the net-

work field into  $N_E$  equal eavesdropping sections, settling one colluding eavesdropper node in the center of each one. Then it eavesdrops the periodic *Ptype* messages heard by them in every maintenance period interval. Suppose the network field is  $1200m \times 600m$  with  $N = 200$  nodes and half of them belong to the multicast group, uniformly distributed. The transmission range of each adversarial node is 150m. Briefly, using conditional Shannon entropy we can calculate the adversary's uncertainty about the leader location as follows. Let's  $H(L|S)$  denote the uncertainty of the adversary to guess where the leader is located, where  $L$  is the random variable of leader's location and  $S$  is a random variable with uniform probability distribution function of  $P(S = section_j) = \frac{1}{N_E}$  ( $1 \leq j \leq N_E$ ) (the probability that the group leader is located in  $section_j$ ).  $P(L_i)$  is the probability that the leader is the node located at location  $L_i$ , while  $1 \leq i \leq N$  when no node is compromised by the adversary and  $1 \leq i \leq N(1 - C/N)$  when  $C$  nodes are compromised.

$$\begin{aligned}
H(L|S) &= - \sum_{j=1}^{N_E} \frac{1}{N_E} \sum_{i=1}^{N(1-C/N)} P(L_i, S_j) \log_2(P(L_i|S_j)) \\
&= - \sum_{j=1}^{N_E} P(S = section_j) \sum_{i=1}^{N(1-C/N)} P(L_i|S_j) \log_2(P(L_i|S_j)) \\
&= - \sum_{j=1}^{N_E} \frac{1}{N_E} \sum_{i=1}^{N(1-C/N)} \frac{N_E}{N(1-C/N)} \times \log_2\left(\frac{N_E}{N(1-C/N)}\right) \\
&= - \log_2\left(\frac{N_E}{N(1-C/N)}\right)
\end{aligned}$$

But if we use *Ptype* mechanism and dummy *Ptype* packets the uncertainty increases. Suppose that a *Ptype* message can be a leader message with the probability of  $P_l$  and can be of any other type of *Ptype* packets with the probability of  $1 - P_l$ . Suppose that  $P_{Ptype\_dummy}$  is the probability that a *Ptype* packet is dummy. Then after some (omitted) calculation of joint and conditional entropy we get the following.

$$\begin{aligned}
H(L, Ptype, dummy|S) &= - \log_2\left(\frac{N_E}{N(1-C/N)}\right) \\
&\quad - P_{Ptype\_dummy} \log_2(P_{Ptype\_dummy}) - (1 - P_{Ptype\_dummy}) \\
&\quad \log_2(1 - P_{Ptype\_dummy}) - P_l \log_2(P_l) - (1 - P_l) \log_2(1 - P_l)
\end{aligned}$$



**Figure 2:** Uncertainty about leader's venue increases from the lowest surface to the middle one when *Ptype* mechanism is used ( $P_l = 0.8$ ) and to the highest one when dummy *Ptype* is also used ( $P_{Ptype\_dummy} = 0.1$ )

#### 4. CONCLUSION AND FUTURE WORK

In this proposal we described shortly some of the main ideas of an anonymous multicast routing protocol for MANETs called AnoMul, which extends the identity free unicast routing idea proposed in ANODR from unicast to multicast ap-

plications. In AnoMul we use our message type unification idea to improve the location privacy aspects. We also optimize the mesh maintenance mechanism to reduce the number of reconnecting nodes when a part of the mesh is disconnected. For group sender anonymity we proposed semi-sender group mechanism. We will simulate AnoMul's performance and publish a detailed version of this work in future.

#### 5. REFERENCES

- [1] Jiejun Kong and Xiaoyan Hong. Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 291–302, New York, NY, USA, 2003.

#### APPENDIX A: RDIS

We propose this mechanism to unify the JREQ and JREP packet formats. We change the JREQ format as follows

$\langle RDIS, TTL, seq\#, trapmulticast, onion, PK - 1time \rangle$

$trapmulticast = f_{K_g}\{group\tag, K_{reveal}\}, f_{K_{reveal}}\{group\tag\}$

We add the *TTL* field to JREQ which is not included in ANODR's RREQ. It is set by the JREQ initiator to a random number and is decremented by every forwarding node till reaches zero.  $K_g$  is the group key shared between the group members as mentioned before. So only the group members are able to decrypt the *trapmulticast* and extract the  $K_{reveal}$ . When a *knot* receives a JREQ message it would send the JREP message of the following format to the JREQ initiator along the route:

$\langle RDIS, \{REPLY, K_{seed}\}_{PK-1time}, f_{K_{seed}}(K'_{reveal}, onion) \rangle$

When a node receives a RDIS message it would try to open the  $\{REPLY, K_{seed}\}_{PK-1time}$ , using any valid one time public key it may have. If it succeeds to see the REPLY tag the message is considered as a JREP message, otherwise it is treated as a JREQ. The JREQ sender will compare the received  $K'_{reveal}$  with the original  $K_{reveal}$  to make sure that the JREP comes from a valid group member. Using this method the group members which the node is joining the mesh through, are authenticated. One extra overhead here is due to the cases that JREP messages are forwarded as JREQ by the nodes out of the route because they do not have the suitable one time public keys. Such packets will be forwarded over some hops till the field correspondent to *TTL* of JREQ messages reaches zero (might be a big number). In order to reduce this overhead we put a fake *TTL* field in the JREP message set by the JREP initiator to a small random number. So if the nodes receiving this JREP message are on the discovered route since they have the one time public key they realize that it is the JREP message intended to them. Then they would forward it to the next node on the route, and also replace the *TTL* in JREP message with a new small random number, and when a node located out of the discovered route receives the JREP message it can not open  $K_{seed}$ , so it will treat that as a JREQ message. Due to the small *TTL* in such message, the message would be propagated in the network only for a small number of hops. Additionally, we need to simulate the fixed fields *seq#* and global trapdoor of JREQ messages in JREP packets (in the same position) by some fake ones to make them similar. Otherwise, the adversary would have a clue to distinguish JREQs from JREPs. The next issue is the whole message's length. Since the total length of JREQ and JREP packets are different the adversary may use their size to distinguish between them. We address this problem by adding a random field of the appropriate length to JREQ packets to equalize the route discovery messages' length. The final format of JREQ and JREP messages are as follows.

$JREQ : \langle RDIS, TTL, seq\#, trapmulticast, onion, PK - 1time \rangle$   
 $random \dots \dots \dots field \rangle$

$JREP : \langle RDIS, TTL, seq\#_{Fake}, trapmulticast_{Fake}, \{REPLY, K_{seed}\}_{PK-1time}, f_{K_{seed}}(K'_{reveal}, onion) \rangle$