Open access • Proceedings Article • DOI:10.1109/AINA.2006.104

ARM: Anonymous Routing Protocol for Mobile Ad hoc Networks — Source link

Stefaan Seys, Bart Preneel

Institutions: Katholieke Universiteit Leuven

Published on: 18 Apr 2006 - Advanced Information Networking and Applications

Topics: Optimized Link State Routing Protocol, Ad hoc wireless distribution service, Wireless Routing Protocol, Vehicular ad hoc network and Wireless ad hoc network

ARM: Anonymous Routing Protocol for Mobile Ad hoc Networks*

Stefaan Seys and Bart Preneel
K.U.Leuven, Department Electrical Engineering-ESAT, SCD/COSIC
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
{stefaan.seys,bart.preneel}@esat.kuleuven.be

Abstract

Due to the nature of radio transmissions, communications in wireless networks are easy to capture and analyze. Next to this, privacy enhancing techniques (PETs) proposed for wired networks such as the Internet often cannot be applied to mobile ad hoc networks (MANETs). In this paper we present a novel anonymous *on demand routing scheme for MANETs. We identify a number of problems of previously proposed works and propose an efficient solution that provides anonymity in a stronger adversary model.*

1. Introduction

Compared to wired networks, MANETs are more vulnerable to both active and passive attacks. Wireless transmissions are easy to capture remotely and undetected, while the lack of central management and monitoring make network nodes susceptible to active attacks. Providing security for MANETs is a challenging task, and many researchers have engaged in designing protocols for diverse security related task such as key management, authentication, confidentiality, etc. Recently researchers have also tackled the problem of anonymity in wireless networks (see related work in Sec. 2). It is clear that providing anonymity in ad hoc networks is important as users may wish to hide the fact that they are accessing some service or communicating with another user. Another application is hiding the location of users participating in the network. Hiding nodes that participate in the network also makes it more difficult for an adversary to focus his attack as he will not be able to identify and locate the more active nodes within the network.

In this paper we describe an anonymous on demand routing protocol for MANETs that is secure against both nodes that actively participate in the network and a passive global adversary that monitors *all* network traffic.

*This work was supported by Concerted Research Actions GOA-Mephisto 2000/06 and GOA-Ambiorix 2005/11 of the Flemish Government, and by the Interdisciplinary institute for BroadBand Technology (IBBT) of the Flemish Government.

2. Related Work

The authors of [1, 3, 7, 8] present a different but similar means of anonymizing mobile ad hoc on demand routing protocols such as DSR [2] and AODV [5]. The main idea is that the source sends a Route Request (RREQ) message targeted at the destination in order to discover one or more routes to this destination. Only the destination can recognize that this RREQ was targeted at it, all other nodes can only verify that it was *not* targeted at them, but no other information is released to them. In SDAR [1] all intermediate nodes add an encrypted version of their identity to the RREQ before forwarding it. Only the destination is able to decrypt the identities collected in the RREQ. The destination uses these collected identities to create a Route Reply (RREP) message that will be returned to the source of the RREQ. In ANODR [3] each intermediate node adds sufficient information to a "boomerang onion" that is copied by the destination into the RREP message. Nodes can recognize RREP messages they need to forward using this boomerang onion. Finally, in ASR [8] and MASK [7], nodes forwarding a RREQ message keep state information about this RREQ. Later on, when they receive a RREP message, they use this state information to decide whether and to whom they have to forward this RREQ.

3. Contributions of our Work

Our work improves on a number of drawbacks of the protocols proposed in [1, 3, 7, 8]. Our proposal also remains secure in a stronger adversary model.

The main drawback of ASR and ANODR is efficiency:

1. Every forwarding node has to generate a fresh public/secret key pair for every RREQ message it forwards. As these RREQs are flooded over the entire network, every node in the network needs to generate a fresh key pair for every RREQ that is released in the network. The private key is stored in the node's routing table. Note that the cost of generating public/private key pairs is non-negligible.
2. RREP messages carry no identifier that can be linked to the RREQ messages. This means that for a node to decide whether it has to forward a RREP or not, it has

to try to decrypt it with every private key it has stored in its routing table.
3. The destination is identified using a trapdoor identifier of the following form: $k_{SD}[\text{dest}]$ where dest is a public binary string that indicates that "you are the destination if you see this" and k_{SD} is a shared key between the source and destination. When a node receives a RREQ it will have to decrypt the trapdoor identifier with *every* key it shares with other nodes.

The main drawbacks of SDAR are:

1. Route discovery is very inefficient as it requires every forwarding node to perform a public key decryption, a public key encryption and a signature generation for every RREQ message it forwards. Again this is a substantial cost as RREQ messages are flooded over the entire network.
2. The destination learns the identity of all the forwarding nodes on the path.
3. In SDAR it is assumed that every broadcast contains the identity of the broadcasting node in plaintext.

The main drawbacks of MASK are:

1. The final destination is contained within every RREQ message in plaintext.
2. MASK relies on a tight synchronization of keys and pseudonyms between neighboring nodes.

Another drawback common to ANODR, ASR, SDAR and MASK is that a powerful passive adversary that monitors all network traffic (without knowing any keys) can trace the RREP message from the destination back to the source. This is because only the intermediate nodes that are assumed to forward the RREP actually do so and all other nodes drop the RREP packet. The authors assume that a global passive adversary cannot link the RREP packets from one hop to the other because they will have a different *appearance*, we argue that a global passive adversary can still trace the RREP packets by following the *flow* of RREP packets (if a RREP enters a node and a fraction of time later a RREP leaves this node, this node is most likely on the path). Exactly the same argument holds for all the DATA packets that are transmitted over the anonymous route between source and destination. This means that the adversary is able to correlate educated guesses on the path and quickly discover source-destination relationships by observing the message *flows*.

The protocol we propose in this paper solves the problems related to anonymity while also solving some of the efficiency problems of the previous proposals.

4. Adversary Model and Goals

In this paper we assume two distinct adversaries. The first adversary is an *external global passive adversary* who can observe all possible communications between all nodes in the network at all time. The goal of our protocol towards this adversary is to (1) prevent him from learning the destination of these messages, and (2) prevent him from learning which nodes are part of the path from the source to the destination.

The second adversary we assume is a *cooperating node inside the network*. This means that we assume that every node that is part of the network is a potential adversary. The goals of our protocol towards this adversary are (1) a node should not be able to determine whether another node in the network is the sender or the destination of a particular message, (2) a node should not be able to determine whether another node is part of a path between two nodes.

5. Our Proposal: ARM

5.1. Assumptions

We assume that every node in the network has a permanent identity that is known by the other nodes in the network that wish to communicate with this node.

Next, we assume that the source S and the targeted destination D share a secret key k_{SD} and a secret pseudonym. The source will include this pseudonym in RREQ messages targeted at this specific destination. The destination will have a list of its pseudonyms (used by different sources) in memory such that he can quickly verify whether a message is targeted at it or not. This pseudonym can only be used once (i.e., for a single RREQ). Different mechanisms can be used to synchronize the pseudonyms used between source and destination (for example an encrypted synchronized counter). The destination needs to store two consecutive pseudonyms, the Nym_i that is currently used and the next Nym_{i+1}. The destination advances this window when it receives a RREQ identified with Nym_{i+1}. In order for our protocol to be efficient, we assume that nodes will only share secret keys and pseudonyms with a limited set of other nodes.

Next, we assume that every node has established a broadcast key with its 1-hop neighborhood. This broadcast key will be used to encrypt the RREP messages. For static networks, nodes can establish these broadcast keys using some random pseudonym and hence stay anonymous towards their neighbors. For dynamic networks with mobile nodes, the key management scheme by Seys and Preneel [6] can be used. Seys and Preneel propose a dynamic key management scheme that periodically established new link keys and broadcast keys with new nodes that enter a node's neighborhood. Again nodes can use pseudonyms while running this key management protocol in order to hide their real identity to the other nodes in the network. By using a fresh pseudonym every time they run the key update protocol nodes can obtain a certain degree of anonymity towards other nodes in the network as they move around. Another possibility is to use the scheme proposed in [7], adapted to support broadcast keys instead of point-to-point keys.

We further assume that wireless links between nodes are symmetric.

5.2. Route Discovery

Assume that source node S wishes to discover a route to the destination D. Nodes S and D share the secret key k_{SD} and D will recognize its current pseudonym Nym_{SD}.

First, S generates a fresh asymmetric key pair $\mathrm{priv}_D/\mathrm{pub}_D$ and a secret key k. Next, S generates the trapdoor identifier $\mathrm{id}_{\mathrm{dest}}$ that can only be opened by node D that has knowledge of the secret key k_{SD}:

$$\mathrm{id}_{\mathrm{dest}} = k_{SD}[D, k, \mathrm{priv}_D], k[\mathrm{Nym}_{SD}] .$$

The last part $k[\mathrm{Nym}_{SD}]$ is later on used to proof that a RREP actually comes from the intended destination D. Next, S generates a random pair of link identifiers (n_S, k_S) that will later on be used to recognize RREP messages. Finally, S encrypts[1] the pair of link identifiers with pub_D and broadcast the following RREQ message (Nym_{SD} also serves as a unique identifier for this RREQ message):

$$S \longrightarrow * \quad : \quad \mathrm{Nym}_{SD}, \mathrm{ttl}, \mathrm{pub}_D, \mathrm{id}_{\mathrm{dest}}, \mathrm{pub}_D(n_S, k_S) .$$

Each node N_i that receives a RREQ message first checks whether it is the targeted destination of the received RREQ by verifying whether Nym_{SD} is in its current list of valid pseudonyms. If so, N_i tries to decrypt the trapdoor identifier $\mathrm{id}_{\mathrm{dest}}$ and verifies whether the first part of the decryption is equal to its global identifier N_i. If this fails, the node was not the targeted destination. Independent of the outcome, the node checks whether Nym_{SD} has been recorded in its routing table. If yes, the node discards the RREQ message. Otherwise the node performs the following tasks, depending on whether it was the targeted destination or not.

If N_i *is not the targeted destination*, it first checks the $\mathtt{ttl}$ field. If $\mathtt{ttl} \leq 1$ then the node decrements the $\mathtt{ttl}$, and generates a random pair of link identifiers (n_i, k_i), appends these to the already received encrypted link identifiers and encrypts everything with pub_D. Finally, N_i stores $(\mathrm{Nym}_{SD}, n_i, k_i, k[\mathrm{Nym}_{SD}])$ in its routing table and broadcasts the following RREQ message:

$$\begin{aligned} N_i \longrightarrow * \quad : \quad & \mathrm{Nym}_{SD}, \mathrm{ttl}, \mathrm{pub}_D, \mathrm{id}_{\mathrm{dest}}, \\ & \mathrm{pub}_D(\ldots(\mathrm{pub}_D(n_{i-1}, k_{i-1}), n_i, k_i) . \end{aligned}$$

If N_i *is the targeted destination*, it performs the same steps as if it was not the targeted destination, but fills the link identifier field with random data of appropriate length (in stead of the encrypted link identifiers):

$$D \longrightarrow * \quad : \quad \mathrm{Nym}_{SD}, \mathrm{ttl}, \mathrm{pub}_D, \mathrm{id}_{\mathrm{dest}}, [\text{random bits}] .$$

After it has forwarded the RREQ message, the destination can prepare its reply message.

We apply both random **padding** and **time-to-live** values for the RREQ messages. We provide a detailed discussions for both in Sec. 6.1 and Sec. 6.2.

[1] $\mathrm{pub}_D(m)$ is the encryption with an asymmetric cipher of message m using the public key of node D.

5.3. Route Reply

After successful decryption of the trapdoor identifier $\mathrm{id}_{\mathrm{dest}}$, the destination has knowledge of k and priv_D. Using priv_D, node D can decrypt the link identifiers that are contained within the received RREQ. With these pairs of link identifiers (k_i, n_i) for $1 \leq i \leq n$ and (k_S, n_S), node D constructs a route reply *onion* of the following form:

$$k_n \Big[n_n, k'_n, k, k_{n-1} \big[n_{n-1}, k'_{n-1}, k, \ldots k_S[n_S, k] \big] \Big] ,$$

with $k'_n = h[k_{n-1}]$ where h is a cryptographic hash function. After the construction of the reply onion, node D generates a random $\mathtt{ttl}$ and broadcasts the following RREP message (Nym_{SD} is copied from the RREQ message and again serves as a unique identifier):

$$* \longleftarrow D \quad : \quad k_{D*}[\mathrm{Nym}_{SD}, \mathrm{ttl}], \textit{onion}$$

The identifier and ttl field (i.e. the header) of the RREP message are encrypted with the current broadcast key k_{D*} of node D to hide them from a global passive adversary.

Each node N_i that receives a RREP message will perform one of two actions (a) or (b) (see below). First it verifies whether it has forwarded a *Route Request* with identifier Nym_{SD}. If not, it proceeds with action (b). Otherwise, the node checks whether it already received a *Route Reply* with this identifier. If so, it again proceeds with action (b). If this is the first time that it sees a RREP that has the same identifier as a RREQ it forwarded earlier, it decrypts the reply onion using k_i and checks whether the first part of the decryption is equal to n_i. If so, node N_i is on the anonymous route. Node N_i now validates the proof k by verifying that the decryption of $k[\mathrm{Nym}_{SD}]$ using k retrieved from the RREP is equal to Nym_{SD}. If the proof fails, the message is discarded. Note that node N_i uses Nym_{SD} to retrieve $(n_i, k_i, k[\mathrm{Nym}_{SD}])$ from its routing table. If the proof is valid, node N_i proceeds with action (a).

(a) Node N_i strips one layer of the reply onion, generates a new random $\mathtt{ttl}$ and broadcasts the RREP message, encrypting the new header with its current broadcast key. Node N_i stores the secret keys $(k'_i, h(k_i))$ in its routing table. The first element $k'_i = h(k_{i-1})$ is a secret it shares with the previous hop in the route, while the second element $h(k_i)$ is a secret it shares with the next hop in the route. We will use the notation k_{prev} and k_{next} respectively in the rest of this paper to denote these keys.

(b) Node N_i replaces the reply onion by random data of appropriate length, decrements the $\mathtt{ttl}$ and broadcasts the RREP message, encrypting the header with its current broadcast key.

We apply both random **padding** and **time-to-live** values for the RREP messages. We provide a detailed discussions for both in Sec. 6.1 and Sec. 6.2.

5.4. Data Forwarding

Once the source of a RREQ message receives a RREP message with the same identifier Nym_{SD}, it can start sending DATA messages to the destination. Similar to sending RREQ messages, DATA messages will have a one-time identifier attached to them. This identifier allows a node on the route to recognize the fact that it is the next hop and that it should forward the message. Forwarding DATA messages is similar to forwarding RREP messages, using the same ttl scheme, but no padding is required. The one-time identifier is computed using the secret key shared between consecutive hops in the route and the counters c and c' that are incremented per message received or sent on this route. The routing table of a node N_i consists of the following elements:

k_{prev}	k_{next}	$\mathrm{id}_{\mathrm{prev}} = k_{\mathrm{prev}}[c]$	$\mathrm{id}_{\mathrm{next}} = k_{\mathrm{next}}[c']$

Nodes that receive a message that is identified with one of the $\mathrm{id}_{\mathrm{prev}}$'s in their routing table replace this identifier with $\mathrm{id}_{\mathrm{next}}$, reset the ttl field using the scheme in Sec. 6.2, and forward the message. After forwarding the message the nodes increment the counters c and c'. In order to change the appearance of the messages as they traverse the network, the identifier and ttl field are encrypted using the nodes' broadcast keys (similar to the forwarding of RREP messages), while the payload is re-encrypted at every hop using the k_{prev} for decryption and k_{next} for encryption. The length of every data message is fixed (the message is padded at the source if necessary).

Nodes that receive a message with an identifier that does not appear in their routing table replace the identifier and the message payload with a random number, decrement the ttl field and forward the message. Again the message identifier and ttl field are encrypted using the node's broadcast key.

6. Selecting Padding and Time-to-Live Values

6.1. Padding

As mentioned in Sec. 5 we apply padding in order to prevent an adversary from learning the number of hops to the source or destination of the RREQ or RREP messages. Our padding scheme should be effective both against a global passive adversary and each of the nodes inside the network that actively takes place in the routing.

We propose the following padding scheme for **RREQ** messages. The source randomly selects a padding length according to the probability distribution in Fig. 1. Node B that receives a RREQ message from node A with a certain length can now compute the probability that it originated at node A or not. The probability that it did *not* originate at the node A is equal to the surface of the area underneath the probability distribution[2] left from the actual length of the packet (the shaded area in Fig. 1) divided by a normalization factor. We see that using the distribution function we propose in Fig. 1 with high probability a node will select a padding length that has a large area left of it, while choosing a padding length close to zero (providing limited anonymity) is unlikely. Nodes forwarding the RREQ messages do not add any padding.

For the **RREP** messages the source chooses a random padding length with a uniform probability distribution between zero and some maximum value (the padding is appended to the inner layer of the route reply onion). Here the padding is not used to hide the source of the RREP, but rather to hide the length of the RREP. Nodes forwarding a RREP message keep the length of the RREP message constant by adding padding the size of the peeled of layer to the end of the RREP message before forwarding it.

6.2. Time-to-Live

If a ttl field is used in the **RREQ** messages then we propose that the source adds a random value (between zero and some maximum) to the required ttl size in order to reach the destination. The ttl is randomized because otherwise it will reveal the distance between the source and the destination.

The ttl field in **RREP** and **DATA** messages is required in order to hide the actual path followed by these messages. Using a fixed ttl value would reveal the path to nodes inside the network that receive RREP or DATA messages from their neighbors (as only nodes on the path will set the ttl value to this fixed value and all other nodes decrement this value). We propose the following padding scheme to prevent this. Every node on the path chooses a ttl value according to the probability distribution in Fig. 2. This distribution has two important properties: (1) small ttl values are favored, and (2) the probability rapidly decreases to reach zero at some maximum ttl value $\mathrm{ttl}_{\mathrm{max}}$. Similar to the discussion on RREQ padding lengths, node B that receives a RREP or DATA message from node A with a certain ttl value can compute the probability that it originated at node A or not. The probability that it did *not* originate at the node A is equal to the surface of the area underneath the probability distribution at the right side of the ttl value it received (the shaded area in Fig. 2) divided by a normalization factor. We see that using the distribution function we propose in Fig. 2 with high probability a node will select a ttl value that has a large area right of it, while choosing a large ttl value (providing limited anonymity) is unlikely. We set a minimum ttl value in order to hide the path to a global passive adversary. Nodes receiving RREP or DATA messages that are not part of the path decrement the ttl before forwarding the message as described in Sec. 5.3 and Sec. 5.4.

7. Analysis of our protocol

7.1. Route Hiding

Our protocol effectively hides routes in the network, both against a passive global adversary and nodes inside the net-

[2]As this is discrete probability distribution, it actually is the sum of all probabilities of selecting a smaller padding length.

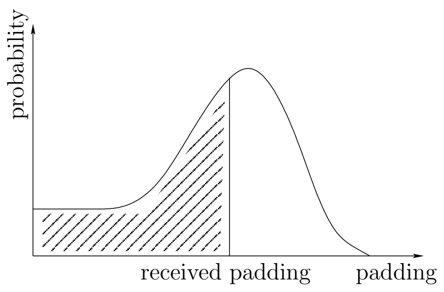

Figure 1. Padding probability distribution.

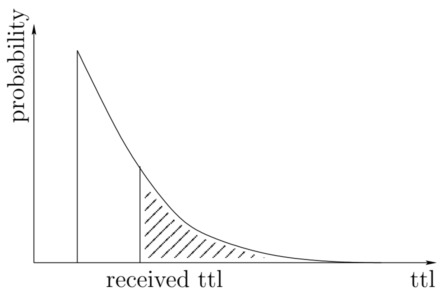

Figure 2. Probability distribution of ttl values.

work. Because of the probabilistic padding and ttl scheme we use, nodes inside the network will not be able to determine whether the node they received a message from is the source of this message or forwarding it (a more detailed analysis of the privacy our random ttl/padding scheme provides will be available in an extended version of this paper). Nor will nodes be able to tell which node is communicating with which other node by inspecting the messages they are forwarding. A passive global adversary will be able to learn which nodes are the sources of fresh messages, but he will not be able to trace this message and learn which nodes are forwarding the message and which node is the final destination. The probabilistic ttl scheme we use hides the actual path between source and destination in a "cloud" of possible paths as this message is forwarded by every node that receives it (whether the node is on the path or not) until the ttl field finally reaches zero.

Note that previous works propose the use of dummy traffic and local mixing of messages to prevent traffic analysis. We argue that these are generic techniques that can be used in any situation to hinder traffic analysis, also in our scheme. The advantage of the limited flooding we propose is that it provides dummy traffic where it is needed (in the vicinity of actual traffic).

7.2. Efficiency

Our protocol requires no cryptographic operations in order for nodes to be able to recognize a message as being targeted at them or not. Next to this, participating in the forwarding of a RREQ message only requires nodes to perform a single public key encryption. When using Rabin or RSA with a small exponent, this can be implemented very efficiently [4]. Our probabilistic ttl scheme makes it possible to hide the path between source and destination in a *localized* cloud without flooding the entire network. Nodes that only participate in the hiding process only need to decrypt and encrypt the short message header using their broadcast keys.

8. Conclusions

Anonymity is an important part of the overall security architecture for mobile ad hoc networks as it allows users to hide their activities. This enables private communications between users while making it harder for adversaries to focus their attacks. In this paper we first identified a number of problems and strengths in previously proposed solutions. We proposed a solution that provides stronger anonymity properties while also solving some of the efficiency problems. We also provide an analysis of how our protocol achieves its goals.

References

[1] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. A novel solution for achieving anonymity in wireless ad hoc networks. In *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 30–38. ACM Press, 2004.

[2] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet draft, IETF MANET Working Group, Apr. 2003. draft-ietf-manet-dsr-09.txt, Work in Progress.

[3] J. Kong and X. Hong. ANODR: anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad hoc Networking and Computing (MobiHoc 2003)*, pages 291–302. ACM Press, 2003.

[4] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[5] C. E. Perkins, E. M. Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. IETF RFC 3561, 2003.

[6] S. Seys and B. Preneel. The wandering nodes: Key management for low-power mobile ad hoc networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems - Workshops (ICDCS 2005 Workshops)*, pages 916–922. IEEE, 2005.

[7] Y. Zhang, W. Liu, and W. Lou. Anonymous communications in mobile ad hoc networks. In *Proceedings of the 24th International Conference of the IEEE Communications Society (INFOCOM 2005)*. IEEE, 2005.

[8] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng. Anonymous secure routing in mobile ad-hoc networks. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004)*, pages 102–108. IEEE, 2004.