

ARM: Anonymous Routing Protocol for Mobile Ad Hoc Networks

S. Seys*

SCD/COSIC,

Department Electrical Engineering (ESAT),

Faculty of Engineering, K. U. Leuven, Belgium

E-mail: stefaan.seys@esat.kuleuven.be

*Corresponding author

B. Preneel

SCD/COSIC,

Department Electrical Engineering (ESAT),

Faculty of Engineering, K. U. Leuven, Belgium

E-mail: bart.preneel@esat.kuleuven.be

Abstract: In this paper we describe a novel anonymous on demand routing protocol for wireless mobile ad hoc networks (MANETs) that is secure against both nodes that actively participate in the network and a passive global adversary who monitors *all* network traffic. Finally, we provided a detailed analysis of the privacy offered by hiding routes in limited broadcast groups, and padding messages.

Keywords: anonymity; routing protocol; ad hoc networks; cryptography.

Reference

Biographical notes: Stefaan Seys received the degree of Master in Electrical Engineering (Burgerlijk Ingenieur Elektrotechniek) from the Katholieke Universiteit Leuven (Belgium) in 2000. In August 2000, Stefaan started working in the research group COSIC (Computer Security and Industrial Cryptography) at the Department of Electrical Engineering (ESAT) of the Katholieke Universiteit Leuven. He completed his Doctorate in Applied Sciences in May 2006. His research interests include cryptography, wireless security and computer security.

Bart Preneel received the Electrical Engineering degree and the Doctorate in Applied Sciences from the Katholieke Universiteit Leuven (Belgium). He is currently full professor at the Katholieke Universiteit Leuven and visiting professor at the T.U.Graz in Austria. He was visiting professor at several universities in Europe. His main research interests are cryptography and information security. He has authored and co-authored more than 200 scientific publications. He is vice president of the IACR (International Association for Cryptologic Research) and a member of the Editorial Board of the Journal of Cryptology and of the IEEE Transactions on Forensics and Information Security. He has participated to several research projects sponsored by the European Commission, for four of these as project manager. He has been program chair of six international conferences (including Eurocrypt 2000, SAC 2005 and ISC 2006) and he has been invited speaker at 20 conferences.

1 INTRODUCTION

The wireless links that are used to connect nodes in a mobile ad hoc network (MANET) are vulnerable to both active and passive attacks. Wireless transmissions are inherently easy to capture remotely and undetected, while the lack of central network management makes ad hoc

networks susceptible to active attacks (e.g., an adversary installing rogue software on one or more victim nodes). Therefore, providing security solutions for ad hoc networks is of primary importance for future large scale deployment.

Many researchers have engaged in designing protocols

Copyright © 200x Inderscience Enterprises Ltd.

for diverse security related tasks, such as key management, authentication, confidentiality, etc. Recently, researchers have also tackled the problem of anonymous routing in ad hoc networks (see related work in Sect. 2). Due to the nature of MANETs, the privacy of the users is at a greater risk than in traditional wired networks such as the Internet. In traditional networks, the amount of traffic a normal user can capture (in addition to his own traffic) is limited to his broadcast domain. For most access technologies (e.g., Asymmetric Digital Subscriber Line (ADSL) or cable) your broadcast domain is limited to your home network. When connected to a Local Area Network (LAN), your broadcast domain is extended to all users connected to this LAN. This means that, in practice, the real privacy threat comes from the Internet Service Providers (ISPs). In MANETs, where mobile nodes exchange data with other nodes without continuous supervision of the user, your “broadcast domain” is unknown and changes continuously. This means that personal data (including the location of a user) now becomes available to many other users that you don’t know or trust.

In this paper we describe a novel 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.

1.1 Adversary model and privacy 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 *corrupted 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.

1.2 Outline

We start by presenting and evaluating related work in Sect. 2. Next, we explain our anonymous routing protocol “ARM” in detail. We describe the use of a trapdoor identifier, the route discovery, route reply and data forwarding protocols. In Sect. 4 we evaluate the anonymity offered by the time-to-live and padding scheme we use in the ARM routing protocol. Finally, in Sect. 5 we give an analysis of the route hiding and efficiency properties of the ARM protocol.

2 RELATED WORK

2.1 ANODR

Kong and Hong (2003) proposed ANODR, one of the first anonymous routing schemes for mobile ad hoc networks. ANODR assumes that the source S and destination D share a secret key k_{SD} . This secret key is used by the source S to generate a trapdoor identifier of the form $E_{k_{SD}}[dest]$ where $dest$ is a public binary string that indicates that “you are the destination”. Obviously, only the destination D who has knowledge of the key k_{SD} can open this identifier. Route Requests (RREQs) have the following format: $\langle seq, E_{k_{SD}}[dest], pub_i, TBO_i \rangle$, with seq a unique sequence number, and TBO (Trapdoor Boomerang Onion) is an onion-like structure of the following form:

$$TBO_i = E_{k'_i} \left[n_i, E_{k'_{i-1}} \left[n_{i-1} \dots E_{k'_S}[src] \right] \right].$$

Here, src is a public binary string that indicates “you are the source”. Every forwarding node N_i adds one layer to the onion using a fresh secret key k'_i and nonce n_i known only to node N_i (see Fig. 1). The forwarding node stores these parameters together with the sequence number in their routing table. Every forwarding node also includes a one-time public key pub_i in the RREQ message before forwarding it (replacing the public key pub_{i-1} of the previous hop). The public key of the previous hop is stored in the routing table. These public keys will be used to setup a secret channel for the future Route Reply (RREP) messages. The source S stores the pair $(E_{k'_S}[src], D)$ in a table in order to recognize future RREP messages it receives in answer to this RREQ message.

The RREP message transmitted by node N_i to node N_{i-1} has the following format: $\langle Pub_{i-1}(k_i), E_{k_i}[TBO_{i-1}] \rangle$. It consists of a fresh link key k_i encrypted with the public key pub_{i-1} of node N_{i-1} . This link key is used to encrypt TBO_{i-1} . Node N_{i-1} uses its private key to decrypt $Pub_{i-1}(k_i)$ contained in the RREP message it received, and uses k_i to decrypt the boomerang onion TBO_{i-1} . It can now verify whether this RREP message was intended for it or not by trying to open TBO_{i-1} using k'_{i-1} and n_{i-1} stored in its RREQ table. It strips away one layer of the boomerang onion, creates the RREP message and transmits it to node N_{i-2} . Node N_{i-1} also stores the link key that was retrieved from the RREP message in its routing table. Note that a RREP message does not contain a unique identifier. The original source S of the route request recognizes itself as the source because it sees the fixed string src after decrypting the received boomerang onion. ANODR uses padding in RREQ and RREP messages to hide the number of hops these messages have traveled.

Data packets are routed using the key k_i shared between two consecutive hops. This key is used to encrypt the payload and to generate a pseudo-random route pseudonym of the form $f^j(k_i)$ for the j th packet that is forwarded.

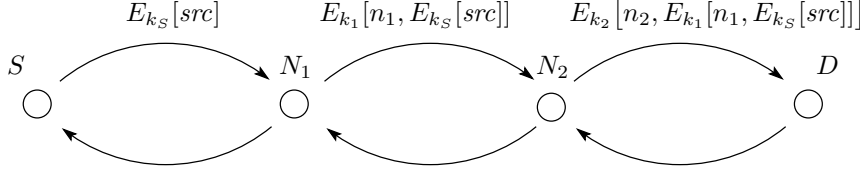


Figure 1: Trapdoor Boomerang Onion used in ANODR. Every node adds a layer to the onion when forwarding RREQ messages. It removes this layer again when forwarding RREP messages.

2.2 ASR

The functionality of the ASR protocol proposed by Zhu *et al.* (2004) is essentially the same as that of ANODR without the use of TBOs. ASR makes no use of onions that are built up as the RREQ progresses through the network, but instead relies on state information that is kept at the forwarding nodes. As in ANODR, every forwarding node includes a one-time public key pub_i in the RREQ message and stores the public key of the previous hop in its routing table.

The RREP message forwarded by node N_i consists of a fresh link key k_i encrypted with the public key pub_{i-1} of node N_{i-1} . This link key is used to encrypt the sequence number contained in the RREQ message. Every node that receives such a RREP message verifies whether it is targeted at it or not using its list of current one-time private keys. Finally, node N_i stores the link key it received from node N_{i+1} and the link key it generated for the next¹ hop N_{i-1} in its routing table.

Data transmission in ASR uses the secret key k_i shared between two consecutive hops on the path. Data packets are encrypted hop-by-hop and are identified using a small TAG. This TAG is of the form $\langle N, MAC_{k_i}[N] \rangle$ with N a non-decreasing number.

2.3 MASK

In MASK (Zhang *et al.* (2005)), nodes use different pseudonyms Nym_i when moving to a new location and establish a shared secret key k_{AB} with each of their neighbors. This key establishment is based on pairing (Barreto *et al.* (2002); Boneh and Franklin (2001)) and is a simple adaptation of the scheme of Balfanz *et al.* (2003) to the mobile setting. Using this shared key, neighboring nodes A and B compute Γ pairs of shared session keys $LinkKey_{AB}^\gamma$ and link identifiers $LinkID_{AB}^\gamma$. Each node keeps a neighbor table in which each entry contains the pseudonym of a neighbor, the pairwise shared session keys and link identifiers ($LinkKey^\gamma, LinkID^\gamma$) and the index γ of the pair that is currently in use. These pairs are used in sequence, i.e., the index γ is increased for every message transmitted and received. New pairs are generated in batches of size Γ as required. A RREQ message contains a unique identifier seq , the identity of the destination D and the current pseudonym Nym_i of the node forwarding or initiating the RREQ message. The forwarding node stores the

pseudonym contained in the RREQ message it received from the previous hop in its reverse route table.

Upon reception of a RREQ message, the destination D prepares a RREP message consisting of the current link identifier $LinkID$ corresponding to the pre-hop-pseudonym contained in the RREQ. The RREP also contains the identity of the destination encrypted with the current session key $LinkKey$ shared between D and the receiving node. Forwarding nodes regenerate this RREP message with their own link keys and identifiers and store the received link identifiers in their forwarding route table. The identity D of the destination is used to link RREQ and RREP messages (i.e., to locate to link identity to be used to forward the RREP message). Finally nodes also keep a table with link identifiers for which they are the final destination.

Data forwarding is based on the link identifiers and corresponding keys in the forwarding route tables. A data packet is reencrypted at every hop.

2.4 SDAR

In contrast to the previously presented protocols, Boukerche *et al.* (2004) do not use temporary or continuously changing identities. Instead SDAR uses a single fixed identity for every node. Every intermediate node inserts its identity as the source address of every message it broadcasts. The source S first generates a fresh public/private key pair $pub_T/priv_T$ and a session key k_{sess} . It uses this session key to encrypt information that will only be disclosed to the destination D : the source identity S , its public key pub_S , the one-time public/private key pair $pub_T/priv_T$, a sequence number seq_S generated by S and finally a signature on all this data. SDAR uses a trapdoor identifier of the form $Pub_D(D, k_{sess})$ which only D can open. A forwarding node N_i simply appends the following information to the RREQ it received: $Pub_T(id_i, k_i, seq_i, Sign(id_i, k_i, seq_i))$. Here id_i is the identity of node N_i , k_i is a fresh link key, and seq_i is a sequence number. The forwarding node stores the sequence number, the link key and the identity of the previous hop (i.e., the node it received the RREQ from) in its RREQ table. The destination opens the trapdoor identifier using its private key and retrieves the session key k_{sess} . With this key D decrypts the third part of the RREQ message and retrieves the one-time private key $priv_T$.

Using this private key, D can now open all data appended by the forwarding nodes and use this data to gener-

¹“Next” in the order of the RREP message.

ate the RREP message. This RREP message has an onion-like structure with the outer-layer decryptable by the last forwarding node, etc. The inner-most layer is decryptable by the source S of the corresponding RREQ message. The layer intended for the source contains all sequence numbers and link keys of all intermediate nodes. As the RREP message traverses the network back to node S , every forwarding node adds the identity of the node it received the RREP message from in its routing table. Random padding in the RREQ messages prevents insiders from learning the number of hops the RREQ messages have traversed.

Data packets are onions generated by the source using the keys and sequence numbers it retrieved from the RREP message. Forwarding nodes strip one layer of encryption and forward the message to the next node according to id_{next} found in its routing table.

2.5 Comparison and evaluation

SDAR is by far the least efficient protocol as it requires both a public key decryption (in order to open the trapdoor identifier), a public key encryption and signature generation every time a node needs to process a RREQ message. ANODR and ASR do not require encryption or decryption, but require that every node that processes a RREQ message generates a fresh public key pair. As RREQs are flooded over the entire network, every node in the network needs to perform these public key operations for every RREQ that is released in the network. In MASK no public key operations are required during route establishment, instead a similar cost is paid because of the continuous process of establishing keys within a node's neighborhood.

ANODR and ASR require a public key encryption and decryption for every RREP a node processes. This cost is multiplied by the number of key pairs in a node's routing table, as the RREP messages contain no identifier that can be linked to the RREQ messages. This means that a node has to decrypt the received RREP message with all private keys in its routing table.

With respect to privacy protection, none of the described protocols offers any protection against an external global passive adversary as this adversary can trivially trace both RREP packets and all consecutive DATA packets as they traverse the network. Although these packets alter their appearance at every hop (because of the onion structure or because they are reencrypted at every hop), the flow of these messages can be traced with high probability. This means that the attacker is able to correlate educated guesses on the path and quickly discover source-destination relationships by observing the message *flows*.

The use of mixing techniques and dummy traffic at every hop will improve the privacy properties of these protocols, but these techniques can be applied independent of the routing protocol that is used and are not considered part of the routing protocol. Moreover, mixing techniques are most effective with large traffic densities. Ad hoc routing protocols often try to achieve a flat routing distribution over all nodes in the network and thus make mixing a less

effective measure.

In the protocol we propose in the next section, we solve all the problems we just mentioned. More specifically, we propose an anonymous routing protocol that allows all participating nodes to recognize messages with a minimum effort, that hides the identity and location of all participating nodes from both insiders and a global adversary, and that shift the largest part of the computational burden to the destination of a RREQ message (instead of spreading in over all potentially participating nodes that forward these RREQ messages).

3 OUR PROPOSAL: ARM

ARM is an on demand routing protocol for MANETs that achieves all the anonymity goals stated in Sect. 1.1 while trying to be as efficient as possible. To achieve this we followed a design strategy that builds on the strengths of the protocols described in Sect. 2 and avoids the weaknesses. We also add new ideas that make the protocol more robust against external global passive adversaries.

3.1 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. These broadcast keys could be established using a similar scheme as the one proposed by Zhang *et al.* (2005), adapted to support broadcast keys instead of point-to-point link keys.

We further assume that wireless links between nodes are symmetric.

3.2 Trapdoor identifier

As we mentioned in Sect. 2, ANODR and ASR use RREQ messages with a trapdoor identifier of the form

$E_{k_{SD}}[dest, x]$, while the trapdoor identifier in SDAR has the form $Pub_D(D, x)$. In both cases x is data that is generated by the source of the RREQ. This results in a performance penalty since every node that receives a RREQ message needs to decrypt this identifier with all keys it shares with other nodes or with its private key.

We propose to use a trapdoor identifier that can be computed by *both* the sender and destination *before* the RREQ message arrives at the destination. This trapdoor identifier can be thought of as a one-time pseudonym Nym_{SD} shared between nodes S and D . Every node keeps a list of pseudonyms it shares with every node it also shares a secret key k_{SD} with. Different mechanisms can be used to synchronize the pseudonym shared between source and destination. One example is the use of a synchronized counter c that is used to compute the valid pseudonym as $Nym_{SD} = E_{k_{SD}}[c]$. This counter is incremented by the source with every RREQ it initiates. 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} . Another example is to assume a synchronized clock between source and destination and use this clock to compute new pseudonyms after a certain time interval in a similar fashion as with the counter. Note that the pseudonym can be shortened as no decryption is required. For example, when using the AES for encryption, the length of the resulting pseudonyms would be 256 bits (the block size of AES). These large pseudonyms can be truncated to fit a sufficient length, for example, 80-bits.

3.3 Route discovery

First, S generates a fresh asymmetric key pair $priv_D/pub_D$ and a secret key k_{pr} . Next, S generates a datagram $info_D$ that can only be opened by node D that has knowledge of the secret key k_{SD} :

$$info_D = E_{k_{SD}}[D, k_{pr}, priv_D, ttl_{init}], E_{k_{pr}}[Nym_{SD}].$$

Here, ttl_{init} is the initial value of the TTL field. Next, S generates a random pair of link identifiers $(n_S, k_S) = (n_0, k_0)$ that will later on be used to recognize RREP messages. Finally, S encrypts the pair of link identifiers with pub_D and broadcasts the following RREQ message (Nym_{SD} also serves as a unique identifier for this RREQ message):

$$S \longrightarrow * : Nym_{SD}, ttl_{init}, pub_D, info_D, Pub_D(n_S, k_S).$$

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 $info_D$ 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.

If N_i is not the targeted destination, the node checks whether Nym_{SD} has been recorded in its routing table.

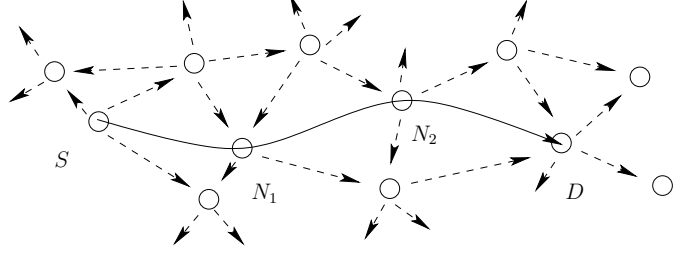


Figure 2: Hidden route from source S to destination D . The full arrow indicates an actual route between S and D , while the dashed arrows indicates fake broadcasts that hide the actual path.

If so, the node discards the RREQ message. Otherwise, N_i stores $(Nym_{SD}, n_{i-1}, k_{i-1}, E_{k_{pr}}[Nym_{SD}])$ in its routing table. If the received $ttl > 1$ then the node decrements ttl , and generates a random pair of link identifiers (n_i, k_i) , appends these to the already received encrypted link identifiers, encrypts everything with pub_D , and broadcasts the following RREQ message (if the received $ttl = 1$, no message is broadcast):

$$N_i \longrightarrow * : Nym_{SD}, ttl, pub_D, info_D, Pub_D(\dots (Pub_D(n_{i-1}, k_{i-1}), n_i, k_i)).$$

If N_i is the targeted destination, it stores the complete RREQ in memory and performs the same steps as if it was not the targeted destination. This means that the destination D behaves exactly the same as all other nodes in the network. After it has forwarded one or more RREQ messages, the destination can prepare its reply message.

3.3.1 Design motivation

The RREQ message is designed with the following goal in mind. *The cost for the intermediate nodes is minimized by moving the bulk of the computational cost to the destination.* This seems only fair as the two communicating end points are the only beneficiaries and the intermediate hops are providing a service to them. Also, the destination can use the Nym_{SD} to identify the source and opt to ignore the RREQ if it is not interested in communicating with the source. When this happens, the cost for the intermediate nodes is minimal. The choice of public key cryptosystem offers two options:

1. By using the Rabin public key cryptosystem, the computation cost for the intermediate nodes is minimized as they are only required to perform a public key *encryption* (Rabin encryption is essentially a modular squaring (Menezes *et al.* (1997))).
2. By using ECIES, the communication cost for the intermediate nodes is minimized as ECIES offers very short cryptograms (320 bits compared to 1024 bits for RSA or Rabin).

3.4 Route reply

Assume that the destination D has collected a number of RREQ messages that were targeted at it. First it decrypts the info_D field contained in one of these RREQs (they all contain the same info_D) and obtains k_{pr} , priv_D and ttl_{init} .

Node D computes the number of hops a RREQ has traversed as $h = \text{ttl}_{\text{init}} - \text{ttl}$. This hop count is useful to select the shortest route and to help with the decryption of the link identifiers when padding is used (see Sect. 3.6).

Using priv_D , node D decrypts the link identifiers (k_i, n_i) for $0 \leq i \leq n$ (with $(k_0, n_0) = (k_S, n_S)$) that are contained within the received RREQs and selects a route it wishes to use (for example based on the hop count). Since the destination itself has also forwarded RREQ messages, it should discard routes that include link identifiers that were generated by D itself in order to ensure loop-free routes.

For every route for which D wishes to generate a RREP message, D repeats the following tasks:

First D generates $n + 1$ link keys s_i for $0 \leq i \leq n$ with $s_0 = s_S$. Link key s_i will be shared between nodes N_i and N_{i+1} . Together with the link identifiers (k_i, n_i) , node D constructs a route reply onion of the following form:

$$\mathcal{O}_n = E_{k_n} \left[n_n, s_n, s_{n-1}, k_{\text{pr}}, E_{k_{n-1}} \left[n_{n-1}, s_{n-1}, s_{n-2}, k_{\text{pr}}, \dots E_{k_S} [n_S, s_S, k_{\text{pr}}] \right] \right].$$

After the construction of the reply onion, node D generates a random ttl (see Sect. 3.6) and broadcasts the following RREP message $(\langle \text{Nym}_{SD}, \gamma \rangle)$ serves as a unique identifier for this RREP message; the index γ is used to distinguish RREP messages that contain the same Nym_{SD} :

$$* \leftarrow D : E_{k_{D*}} [\text{Nym}_{SD}, \gamma, \text{ttl}], \mathcal{O}_n$$

When only a single RREP is generated, the index γ is not required. 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). Note that both actions are implemented in such a way that they both require an equal amount of time (in order to prevent timing analysis). The RREP contains identifier Nym_{SD} and index γ . 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 the same identifier and index. If so, it again proceeds with action (b). If this is the first time that it has seen a RREP with this particular index, 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 of the destination by verifying that the decryption of $E_{k_{\text{pr}}} [\text{Nym}_{SD}]$ (using k_{pr} retrieved from the RREP) is equal to Nym_{SD} . Note that node N_i uses Nym_{SD} to retrieve $\langle n_i, k_i, E_{k_{\text{pr}}} [\text{Nym}_{SD}] \rangle$ from its routing table. If the

proof is valid, node N_i proceeds with action (a); otherwise it discards the message.

- (a) Node N_i strips one layer from the reply onion, generates a new random TTL value ttl and broadcasts the RREP message, encrypting the new header with its current broadcast key. Node N_i stores the secret keys (s_i, s_{i-1}) in its routing table. The first element is a secret key it shares with the next hop N_{i+1} in the route, while the second element is shares with the previous hop N_{i-1} in the route. We will use the notation k_{prev} and k_{next} , respectively, to denote these keys.
- (b) Node N_i replaces the reply onion by random data of appropriate length, decrements ttl and broadcasts the RREP message, encrypting the header with its current broadcast key.

3.4.1 Design motivation

- Onion creation is efficient as it only requires symmetric encryption.
- It is efficient for an intermediate node to check whether an onion is intended for it or not.
- We use a limited broadcast to hide the real route in a cloud of messages. The time-to-live value can be used to select the size of this cloud.
- The time-to-live value is encrypted with a node's broadcast key. This is only necessary to protect against an *external global passive adversary*.

3.5 Data forwarding

Every RREP message with an identifier Nym_{SD} that arrives at the source S contains a route to the destination D . When the RREP messages arrive at the source S , they have the following form:

$$* \leftarrow N_1^\gamma : E_{k_{N_1^\gamma}} [\text{Nym}_{SD}, \gamma, \text{ttl}], E_{k_S} [n_S, s_S^\gamma]$$

Index γ is used to indicate those fields that may differ for the different RREP messages that return in response to a single RREQ message broadcast by the source with identifier Nym_{SD} . Every link key s_S^γ represents a different route to the destination D .

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. 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}	$id_{\text{prev}} = E_{k_{\text{prev}}} [c]$	$id_{\text{next}} = E_{k_{\text{next}}} [c']$
-------------------	-------------------	--	---

Nodes that receive a message that is identified with one of the id_{prev} 's in their routing table replace this identifier with id_{next} , reset the TTL field using the scheme in Sect. 3.6, 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 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.

3.6 Padding and time-to-live schemes

Without randomized padding and time-to-live values, the length of routing messages and the TTL value they contain reveals information to both the global adversary and to inside nodes participating in the routing process. We now propose a possible strategy for both padding and time-to-live value selection. In the next section we analyze the privacy our scheme offers.

3.6.1 Time-to-live

For **Route Request** messages we propose not to use a TTL field for small networks. For large networks, the cost of network-wide broadcast can be cut down by using an safe estimate t_{est} of the hop distance between source and destination. If nodes have a clear idea of the hop distance to other nodes, then they should randomize the TTL value by adding a random value between zero and some maximum to the required TTL size in order to reach the destination ($t_{init} = t_{est} + t_{rand}$ with $0 \leq t_{rand} \leq t_{max}$).

The TTL field in **Route Reply** 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. 4. 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 t_{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 . The probability that it did *not* originate at the node A is related to the surface of the area underneath the

probability distribution at the right side of the TTL value it received. We see that using the distribution function we propose in Fig. 4, with high probability a node will select a TTL value that has a large area to the 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 Sect. 3.4 and Sect. 3.5.

3.6.2 Padding

Without padding the length of a **Route Reply** message would continuously decrease and hence reveal the number of hops the RREP still needs to travel in order to reach its target S . In order to prevent this, node D (the source of the RREP message) adds padding bits to the RREP before transmitting it. The length l of this padding is selected using a uniform probability distribution ($l_{min} \leq l \leq l_{max}$). Nodes forwarding a RREP message keep the length of the message constant by right padding the RREP message before forwarding it to compensate for length reduction created by the peeled off layer.

DATA messages are chopped into packets of equal length. The last packet is padded if necessary.

The length of **Route Request** messages grows as they traverse the network. Without additional padding, the length of a RREQ message discloses the distance the message has traveled. Moreover, the length of a fresh RREQ message broadcast by the source S is known to all nodes in the network (in contrast to the initial length of a Route Reply message, which can vary).

We use a similar design philosophy for the padding scheme for RREQ as we used for the time-to-live scheme for RREP messages. The source randomly selects a padding length according to the probability distribution in Fig. 5. Note that a padding length of 5 means that the source adds random bits such that it seems that the RREQ message has already traveled 5 hops. 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 . The probability that it did *not* originate at the node A is equal to the surface of the area underneath the probability distribution left of the actual length of the packet. We see that using the distribution function we propose in Fig. 5, with high probability a node will select a padding length that has a large area to the left of it, while choosing a padding length close to zero (providing limited anonymity) is unlikely. Note that large padding lengths offer higher privacy at a higher cost (since the message becomes larger and needs to be encrypted at every forwarding hop). Nodes forwarding the RREQ messages do not add any padding.

3.7 Variations

The first variation is to use *no broadcast encryption* for the RREP and DATA packets. The resulting scheme still

hides the real route from insiders, but no longer hides the route from an external global passive adversary. This is because a global adversary can trace the route using the TTL values (all nodes broadcasting packets with the highest TTL value are on the route, the other nodes are just forwarding packets). Because of the limited view of the insiders, they cannot make this analysis.

The second variation is to also drop the use of the limited broadcast of RREP and DATA packets (set the time-to-live to zero). This is obviously the most efficient version, but now routes are also visible to insiders neighboring the route (as only they will receive the DATA and RREP packets).

4 PADDING AND TTL SELECTION

4.1 Privacy offered by random TTL selection

We will now investigate how much privacy is offered by TTL value selection according to a given probability distribution. Suppose that node B receives a message with a TTL value t from its neighbor node A (see Fig. 3). The adversary in this case is node B that has received the message from node A .

Let \mathcal{P} be the discrete probability distribution of the TTL value selection, with values p_t , where p_t is the probability that a node selects an initial TTL value t ($1 \leq t \leq t_{\max}$).

Assume that the adversary has a priori knowledge of the probabilities that some node was the originator of the message that traverses route R . Let \mathcal{S} be this discrete probability distribution with values $s_{N_i}^R$, where $s_{N_i}^R$ is the a priori probability that node N_i is the originator of the message that traverses route R . This probability distribution can vary for every message the adversary observes.

Definition 4.1 (Privacy offered by a TTL value equal to t). *We define the privacy of node A towards node B as the probability that node A was not the originator of the message with TTL value equal to t .*

The probability that a message with a TTL value of t did originate at node A is given by the conditional probability $P(\mathcal{A}|\mathcal{B})$ with:

- \mathcal{A} : node A generates a message with TTL value t ;
- \mathcal{B} : node B receives a message with TTL value t .

Using Bayes' theorem we can rewrite $P(\mathcal{A}|\mathcal{B})$ as $P(\mathcal{B}|\mathcal{A})P(\mathcal{A})/P(\mathcal{B})$. One can easily verify that $P(\mathcal{B}|\mathcal{A}) = 1$. The probability $P(\mathcal{A})$ that A generates a message with TTL equal to t is equal to the product of the probability of A generating the message and the probability that it has a TTL value of t :

$$P(\mathcal{A}) = s_A p_t \text{ with } s_A = \sum_{\forall R} s_A^R. \quad (1)$$

We sum over all possible routes since all messages A sends will be received by its neighbor node B .

The probability $P(\mathcal{B})$ is the probability that some node in the network generated a message that resulted in a TTL value t when it reached node B . This can be expressed as

$$P(\mathcal{B}) = \sum_{N_i, R}^{|N_i B|^R \leq t_{\max} - t} s_{N_i} p_{t+h-1} \text{ with } h = |N_i B|^R. \quad (2)$$

Here, $|AB|^R$ is the hop distance between nodes A and B using route R . Note that the hop distance between two nodes depends on the route R the message follows. The summation goes over all nodes and all routes that originate at these nodes that pass through node A . The hop distance is limited since the TTL value a node can select is limited to t_{\max} .

Using Eq. (1) and Eq. (2) we can compute the privacy offered by a TTL value equal to t as

$$Priv_t = 1 - \frac{P(\mathcal{A})}{P(\mathcal{B})} = \frac{P(\mathcal{B}) - P(\mathcal{A})}{P(\mathcal{B})}. \quad (3)$$

This privacy depends on the probability distributions \mathcal{P} and \mathcal{S} . The former is a system parameter, while the latter is the a priori knowledge the adversary has of the network (number of nodes, possible routes, probability of sending messages, etc.). At design time the a priori knowledge of the adversary is not known and we need to make some assumptions to select the optimal probability distribution \mathcal{P} .

We can now compute the average privacy of a node that generates messages with TTL values selected according to the probability distribution \mathcal{P} as

$$Priv(\mathcal{P}) = \sum_{t=1}^{t_{\max}} p_t Priv_t. \quad (4)$$

Equivalent to the average privacy, we can also compute the average "cost" of a TTL distribution as ($f(t)$ is function that indicates how fast the number of nodes, that will forward a message originating from node A with a TTL value t , grows).

$$Cost(\mathcal{P}) = \sum_{t=1}^{t_{\max}} p_t f(t). \quad (5)$$

The *optimal* distribution for the TTL value selection is the distribution that maximizes $Priv(\mathcal{P})/Cost(\mathcal{P})$.

4.1.1 Selecting \mathcal{P}

In order to be able to evaluate Eq. 4 and Eq. 5, we have to make some assumptions. The first assumption we make is that all nodes are equally likely to generate a message at any time. The second assumption we make is that routes with shortest path length are selected amongst all possible routes. This last assumption means that a node with shortest hop distance h from node B has to generate a message with a TTL value equal to $t + h - 1$ in order for this message to arrive at node B with a TTL value of t .

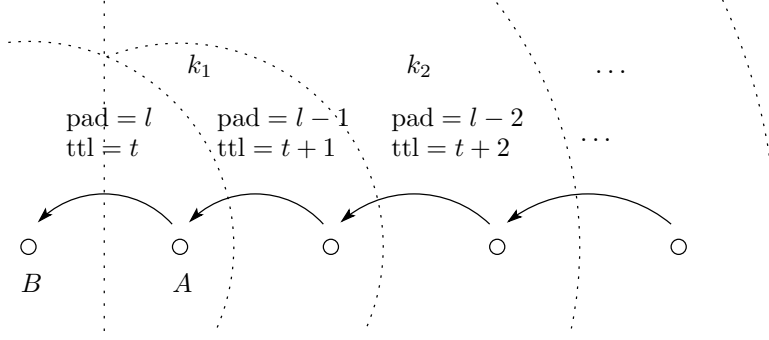


Figure 3: Evolution of padding and TTL values of a message. Node A receives a RREQ message with padding length l , or a RREP or DATA message with TTL value t . This message could have originated from node A or from nodes further away.

This also means that nodes that are closer to B than to node A will never select a route that first passes A and then B . This also means that the message B receives from A could not have originated from a node within B 's range (besides node A , obviously). These assumptions are depicted in Fig. 3. We further assume that the density of nodes is constant throughout the network. This means that the number of nodes k_i at a certain hop distance i grows linearly with i : $k_i = C(2i - 1)$ for $i \geq 1$. This also means that $f(t) = t^2$ in Eq. 4. Note that this is true for nodes that are scattered on a flat surface; if the nodes are located within a three dimensional space, then the number of nodes at a specific hop distance grows quadratically with the distance and $f(t) = t^3$ in Eq. 4.

Using these assumptions, we can evaluate the average privacy and cost offered by a certain probability distribution \mathcal{P} for TTL value selection. Table 1 shows these values for two different classes of probability distributions: exponential and linear. The node density was fixed at $C = 12$ and $t_{\max} = 5$. The normalization factor needs to be chosen such that $\sum_i p_i = 1$. We see that the exponential distribution offers the highest attainable privacy and the highest attainable privacy/cost ratio. For growing values of t_{\max} both the linear and the exponential distribution offer larger maximum privacy values, but at a lower privacy/cost ratio. For the linear distribution, the maximum privacy/cost ratio decreases when t_{\max} grows. In contrast, the exponential distribution offers a constant maximum privacy/cost ratio, independent of t_{\max} . This maximum is attained for a parameter selection $a = 0.22$. The node density C of the network has a positive influence on the privacy performance of the scheme as it lowers the probability that node A was the originator of the message.

Figure 4 shows the probability p_i that a certain TTL value is selected, the privacy $Priv_i$ and the (normalized) cost $Cost_i$ offered by this selection, and the average privacy offered by the exponential probability distribution \mathcal{P} : $p_i = \beta a^i$ with $a = 0.35$. We see that (except for the last couple of TTL values) the privacy offered is independent of the TTL value that is selected. We also see that the most likely TTL values have the lowest cost.

Table 1: Average privacy and privacy/cost ratio of two probability distributions \mathcal{P} for TTL value selection. The numbers in boldface indicate maximal attainable values (β is a normalization factor).

\mathcal{P}	Param a	$Priv(\mathcal{P})$	$\frac{Priv(\mathcal{P})}{Cost(\mathcal{P})}$
Exponential	0.22	0.56	0.28
$p_i = \beta a^i$	0.59	0.82	0.15
Linear	0	0.72	0.066
$p_i = \beta + ai$	-0.1	0.74	0.15
	-0.071	0.81	0.12

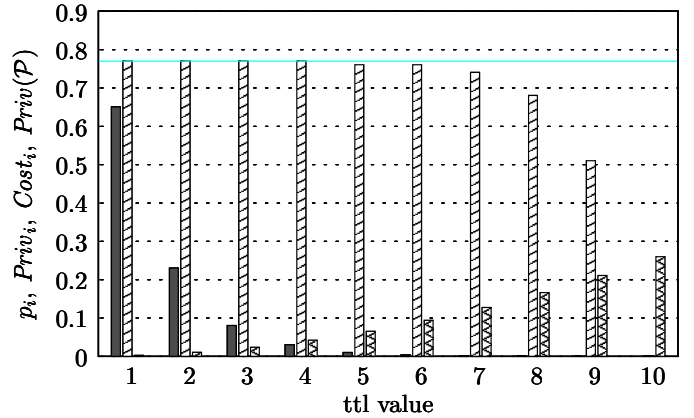


Figure 4: Privacy and cost of an exponential probability distribution for TTL value selection ($p_i = \beta a^i$ with $a = 0.35$). The first bar is the probability a TTL is selected, the second the privacy offered and the third the cost. The horizontal line is the average privacy offered by the distribution.

4.2 Privacy offered by random padding length selection

The evaluation of the privacy offered by random padding selection is very similar to the evaluation of the privacy offered by random time-to-live value selection. Assume the same situation as in Sect. 4.1, but now node B receives a

message from node A with a “length” l . The term “length” indicates how many hops this message seems to have traveled from the source. A message broadcast by the source S without any padding has length $l = 0$. A message that has traveled one hop from the source, or has padding length 1, has a total length of 1, etc. With these definitions, the events \mathcal{A} and \mathcal{B} become:

- \mathcal{A} : node A generates a message with length l ;
- \mathcal{B} : node B receives a message with length l .

Let \mathcal{P} be the discrete probability distribution of the padding length selection, with values p_l , where p_l is the probability that a node selects a padding length l ($0 \leq l \leq l_{\max}$).

Definition 4.2 (Privacy offered by a padding length l). We define the privacy of node A towards node B as the probability that node A was not the originator of the message with padding length l .

Analogous to Sect. 4.1 we can derive equations for $P(\mathcal{A})$, $P(\mathcal{B})$, $Priv_l$, $Priv(\mathcal{P})$, $Cost_l$ and $Cost(\mathcal{P})$. If we assume that the density of nodes is constant throughout the network, then

$$Cost(\mathcal{P}) = \sum_{l=0}^{l_{\max}} p_l(1 + \alpha l).$$

Here, α is the ratio of the true length of a single padding block (in bits) to the length of a RREQ without padding as it leaves the source. The length of a single padding block is approximately 1024 bits when using RSA, while the length of an unpadded RREQ is approximately 3530 bits when using AES and $|D| = 64$, $|k| = 80$, $|Nym_{SD}| = 64$, and $|ttl| = 10$. This results in $\alpha \approx 0.3$. Note that the cost only grows linearly with the padding length l ; this is because this padding is only added once at the source.

Table 2 shows numerical results for exponential and linear probability distributions. The number of 1-hop neighbors is fixed at $C = 6$ and $l_{\max} = 4$. We see that the exponential distribution offers the highest attainable privacy. For growing values of t_{\max} both the linear and the exponential distribution offer larger maximum privacy values, but at a lower privacy/cost ratio. The node density C of the network has a positive influence on the privacy performance of the scheme as it lowers the probability that node A was the originator of the message.

Figure 5 shows the probability p_i that a certain padding length is selected, the privacy $Priv_i$ and the (normalized) cost $Cost_i$ offered by this selection, and the average privacy offered by the exponential probability distribution $\mathcal{P} : p_i = \beta a^i$ with $a = 1/0.35$. We see that (except for the first couple of padding lengths) the privacy offered is independent of the padding length that is selected. We also see that the most likely padding values have the highest cost (in contrast to the TTL scheme). Fortunately, the padding scheme is only used for RREQ messages and not for the more common DATA packets.

Table 2: Average privacy and privacy/cost ratio of two probability distributions \mathcal{P} for padding length selection. The numbers in boldface indicate maximal attainable values (β is a normalization factor).

\mathcal{P}	Param a	$Priv(\mathcal{P})$	$\frac{Priv(\mathcal{P})}{Cost(\mathcal{P})}$
Exponential	1.16	0.77	0.46
$p_i = \beta a^i$	1.70 ($\frac{1}{0.59}$)	0.82	0.44
Linear	0	0.72	0.45
$p_i = \beta + ai$	0.032	0.77	0.46
	0.071	0.81	0.44

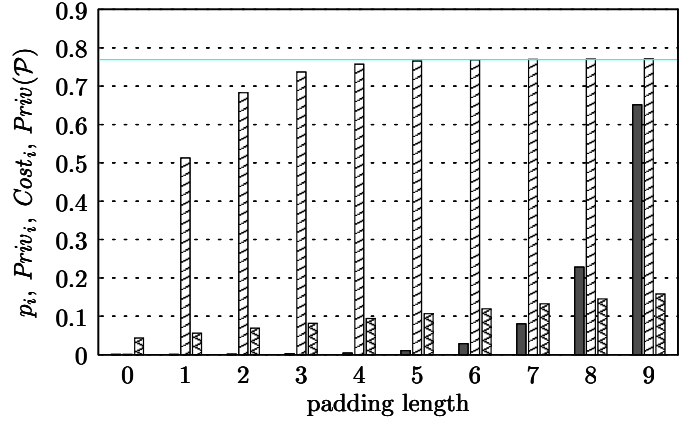


Figure 5: Privacy and cost of an exponential probability distribution for padding length selection ($p_i = \beta a^i$ with $a = 1/0.35$). The first bar is the probability a padding length is selected, the second the privacy offered and the third the cost. The horizontal line is the average privacy offered by the distribution.

5 ANALYSIS OF THE PROTOCOL

5.1 Route Hiding

Our protocol effectively hides routes between sources and destinations, both against a passive global adversary and nodes inside the network. 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. Nor can they tell which nodes are part of a route between two nodes. Nodes on a route are not 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 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 (see Fig. 2). Note that this scheme

only works in networks where every node is surrounded by multiple neighbors. In very sparse networks, hiding routes is very difficult, as there are only a limited number of possible routes an actual message could have followed.

5.2 Efficiency

Our protocol requires no cryptographic operations in order for nodes to be able to recognize whether a message is 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, this can be implemented very efficiently (Menezes *et al.* (1997)). Our probabilistic TTL scheme makes it possible to hide the path between source and destination 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.

6 CONCLUSIONS

Anonymity is an important part of the overall security architecture for mobile ad hoc networks. Without sufficient measures to protect the privacy, users will leave a digital trace of all their actions (including their location). In this paper we first give an analysis of the current state of the art in anonymous ad hoc routing protocols and indicate the strengths and weaknesses. We proposed a novel anonymous routing protocol, clearly indicating our design motivations. Finally, we provided a detailed analysis of the privacy offered by hiding routes in limited broadcast groups, and padding messages.

ACKNOWLEDGMENT

This work presented in this paper was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and in part by the Interdisciplinary institute for BroadBand Technology (IBBT) of the Flemish Government.

REFERENCES

- D. Boneh and M. K. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology - CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer-Verlag, 2001.
- 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*, pp. 30–38, ACM Press, 2004.
- 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)*, pp. 291–302, ACM Press, 2003.
- A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.
- Y. Zhang, W. Liu, and W. Lou, "Anonymous communications in mobile ad hoc networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, vol. 3, pp. 1940–1951, IEEE, 2005.
- B. Zhu, Z. Wan, M. S. Kankanhalli, F. Bao, and R. H. Deng, "Anonymous secure routing in mobile ad-hoc networks," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004)*, pp. 102–108, IEEE, 2004.
- D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H. C. Wong, "Secret handshakes from pairing-based key agreements," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pp. 180–196, IEEE, 2003.
- P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Advances in Cryptology - CRYPTO 2002*, vol. 2442 of *Lecture Notes in Computer Science*, pp. 354–368, Springer-Verlag, 2002.