Node Cooperation in Hybrid Ad Hoc Networks

Naouel Ben Salem, *Student Member*, *IEEE*, Levente Buttyán,
Jean-Pierre Hubaux, *Senior Member*, *IEEE*, and Markus Jakobsson

Abstract—A hybrid ad hoc network is a structure-based network that is extended using multihop communications. Indeed, in this kind of network, the existence of a communication link between the mobile station and the base station is not required: A mobile station that has no direct connection with a base station can use other mobile stations as relays. Compared with conventional (single-hop) structure-based networks, this new generation can lead to a better use of the available spectrum and to a reduction of infrastructure costs. However, these benefits would vanish if the mobile nodes did not properly cooperate and forward packets for other nodes. In this paper, we propose a charging and rewarding scheme to encourage the most fundamental operation, namely packet forwarding. We use "MAC layering" to reduce the space overhead in the packets and a stream cipher encryption mechanism to provide "implicit authentication" of the nodes involved in the communication. We analyze the robustness of our protocols against rational and malicious attacks. We show that—using our solution—collaboration is rational for selfish nodes. We also show that our protocols thwart rational attacks and detect malicious attacks.

Index Terms—Network-level security and protection, wireless communication, authentication security, payment schemes.

1 Introduction

The geographic area covered by a conventional structure-based network (e.g., cellular network, WiFi network, etc.) is populated with base stations (also called access points) that are connected to each other via a backbone. A mobile node can use the network when it has a direct (single-hop) connection to a base station, but as soon as it is beyond the reach of the base stations' coverage, the mobile node is disconnected from the structure-based network. For the operator, the usual solution to this problem consists in increasing the coverage by adding antennas and for the user to move until he reaches a covered region. An alternative solution[1] would be to allow multihop communications in the structure-based network, which would make it possible for the isolated node to ask other nodes to relay its traffic to or from a base station.

The resulting hybrid ad hoc network [1], [27], [11], [3], [25], also called *multihop cellular network*, offers several benefits [18], [19]. First of all, the coverage of the network is increased while the number of fixed antennas is kept relatively small. Reducing the number of antennas is beneficial for the operator because it represents a cost reduction and also because of the "NIMBY" (Not in my back yard) [24] attitude that makes site acquisition and approval both tedious and difficult. Second, the energy consumption of the nodes can be reduced because the signal has to cover a smaller distance. And finally, as the radiated energy is reduced, the interference with other nodes diminishes as well.

Given the advantages listed above, hybrid ad hoc networks represent a new and promising paradigm. However, the proper operation of this new family of networks requires the mobile nodes to collaborate with each other. This collaboration cannot be taken for granted in a civilian network because each user wants to maximize his benefit while minimizing his contribution. Indeed, forwarding packets is energy-consuming and a selfish user can tamper with his mobile device to remove the relaying functions or simply shut down the device when he is not using it. A systematic denial of the packet forwarding service would remove all the benefits introduced by the multihop aspect of the communications.

In this paper, we propose a set of protocols to foster cooperation for the packet forwarding service in hybrid ad hoc networks. This solution is based on a charging and rewarding system.

This paper extends and completes our previous treatment of the same problem [4]. This work is part of the MICS Terminodes Project [14]. The rest of the paper is organized as follows: We introduce the system, including the adversarial model, in Section 2 and describe our proposed protocols in Section 3. In Section 4, we analyze the robustness of our solution against rational and malicious attacks and we show that the charging and rewarding scheme encourages cooperation in hybrid ad hoc networks. In Section 5, we present an estimate of the communication and computation overhead of our protocols. Finally, we describe the related work in Section 6 and we present our conclusions and future work in Section 7.

1. Note that we do not assume that multihop communication is always the best solution to increase infrastructure coverage. The decision whether or not a given network should be extended using multihopping is out of the scope of this paper.

• *N. Ben Salem and J.-P. Hubaux are with the Laboratory of Computer Communications and Applications (LCA), EPFL—Lausanne, Batiment BC (BC200), Station 14, CH-1015 Lausanne, Switzerland.
E-mail: {naouel.bensalem, jean-pierre.hubaux}@epfl.ch.*
• *L. Buttyán is with the Department of Telecommunications, Budapest University of Technology and Economics, BME-HIT, PO Box 91, 1527 Budapest, Hungary. E-mail: buttyan@hit.bme.hu.*
• *M. Jakobsson is with the School of Informatics, Indiana University at Bloomington, Bloomington, IN 47406. E-mail: markus@indiana.edu.*

*Manuscript received 15 July 2003; revised 24 Sept. 2004; accepted 18 Nov. 2004; published online 16 Feb. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0103-0703.*

2 System Model

2.1 Assumptions

The system consists of a set of *base stations* connected to a high speed backbone and a set of *mobile nodes*. The mobile nodes use the base stations and, if necessary, the backbone to communicate with each other or with a host connected to the backbone. Communication between the mobile nodes and the base stations is based on wireless technology and the nodes are loosely synchronized with their base station. We assume that all communication is packet-based and that all the base stations and the backbone are operated by a *single operator* that is fully trusted by all mobile nodes, be it for charging, for route setup, or for packet forwarding. For the sake of simplicity, we consider that the nodes and the base stations have the same power range, which, we assume, will lead to bidirectional links (i.e., even if the quality of the link is not necessarily the same in both directions, we assume that the communication is still possible in both directions).

We call a *cell* [18] the geographical area that is controlled by a given base station. The power range of the base station is smaller than the radius of the cell, meaning that some nodes have to rely on *multihop relaying* to communicate with the base station. We consider a model in which the nodes move. However, we assume that the routes are stable enough to allow for the sending of a substantial number of packets and, thus, to amortize the cost of running a routing protocol (see Section 5). We assume each node i to be registered with the operator and to share a long-term symmetric key K_i with it. K_i is the only long-term cryptographic material stored in i. The secret keys of all the nodes in the network are maintained at the operator.

2.2 Rationale of the Solution

When a mobile node A (the initiator) wants to communicate with another mobile[2] node B (the correspondent), it first establishes an *end-to-end session* with B. As we will see in detail in Section 3.2 a session is a route on which all nodes are authenticated. This is done by establishing an *initiator session* between A and the base station of the initiator BS_A and a *correspondent session* between the base station of the correspondent BS_B and B. These sessions are used to exchange packets between A and B, in both directions.

For each packet, we call S its source (which is A or B) and D its destination (therefore, B or A, respectively). The base stations of S and D are denoted by BS_S and BS_D, respectively. The packet is then sent by the source S to BS_S, if necessary in multiple hops. If D resides in a different cell, then the packet is forwarded by BS_S to BS_D via the backbone. Finally, the packet is sent to D, possibly in multiple hops again. If one of the routes is broken, then a new session is established using an alternative route. Note that the system model described above is similar to that of [18] with the difference that we require *all* communication to pass through a base station. Although this may lead to suboptimal routes, our model has the advantage of significantly reducing the complexity of routing from the nodes' point of view since they have to maintain only a single route (to the base station) instead of one route per correspondent. Of course, the base station has to maintain a route to every node in its cell.

2. We consider mobile-to-mobile communication as it is the most complete case.

To encourage the intermediate nodes to forward the traffic, we propose to charge the initiator A for the traffic in both directions and to reward the forwarding nodes (the operator is rewarded as well). We take advantage of the presence of the trusted operator and assume that it maintains a billing account for every node in the system; our remuneration scheme (see Section 3.4.1) is implemented by manipulating the appropriate billing accounts.

Our protocols are based entirely on symmetric key cryptography. Although asymmetric cryptographic primitives may seem to be more suitable for implementing some of the functions of our scheme, they have a high computational overhead (compared to symmetric key primitives), which prevents their application in resource constrained mobile devices.

2.3 Adversarial Model

Attacker model. An attacker $\mathcal{M}$ is *rational* if it misbehaves only when this is beneficial in terms of remuneration, service provision or saving resources. Otherwise, $\mathcal{M}$ is *malicious*. The users are selfish and, thus, each node in the network is potentially an attacker. We assume that several attackers can collude to perform more sophisticated attacks. We also assume that an attacker is occasionally able to compromise "good" nodes by retrieving their secret keys.

Attack Model. We do not attempt to ensure secrecy or anonymity of communication and, thus, we do not study *passive* attacks (where the attacker analyzes the data without altering it). Instead, we are interested in *active* attacks, where the attacker modifies, deletes, or injects data in the network. We consider exclusively the attacks performed against our solution (e.g., we do not consider DoS attacks based on jamming) and we identify the following active attacks:

- *Packet dropping*: $\mathcal{M}$ drops a packet it is asked to forward.
- *Replay*: $\mathcal{M}$ replays a valid packet from an expired or still existing session.
- *Filtering*: $\mathcal{M}$ modifies a packet it is asked to forward.
- *Emulation*: $\mathcal{M}$ uses the secret key of a node it compromised to perform actions in its name.

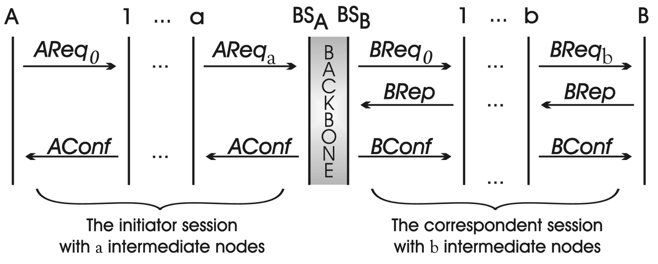
2.4 Interaction with the Underlying Routing Protocol

Our solution assumes the existence of an underlying (proactive or reactive) ad hoc routing protocol that provides the initiator A and the base station BS_B with the *initiator route* (route between A and BS_A) and the *correspondent route* (route between BS_B and B), respectively. The main incentive for the nodes on these routes to cooperate in the routing is the expected future benefit (i.e., the remuneration). Our solution does not require the underlying routing protocol to be secure. Indeed, the operator is able, in our solution, to detect several routing attacks such as those described in [13] (see Section 4.6 for more details).

3 Proposed Solution

3.1 Building Blocks and Notation

Our protocols use two cryptographic building blocks: A MAC (Message Authentication Code) function and a stream cipher [21]. However, our use of these primitives is unconventional:



A
1 ... a
BSA BSB
1 ... b
B
AReq0
AReqa
BACKBONE
BReq0
BReqb
BRep
BRep
AConf
AConf
BConf
BConf
The initiator session with a intermediate nodes
The correspondent session with b intermediate nodes

Fig. 1. The session setup phase.

- During the session setup phase (see Section 3.2), we need all the nodes in the path to authenticate the request message and, instead of appending one MAC computed by each of the nodes to the message, we use an iterative "MAC layering" technique. The principle of this technique is explained in Section 3.2. Our solution achieves a similar effect to that of the classical MAC appending technique but keeps the size of the request constant. Therefore, our technique is more efficient in terms of bandwidth usage. To the best of our knowledge, such a scheme has not been proposed yet for ad hoc networks.
- During the packet sending phase (see Section 3.3), we apply an iterative stream cipher encryption mechanism that can be considered as an "implicit" authentication mechanism because it allows the operator to verify that the packet took the route it was supposed to take. At the same time, it thwarts the free-riding attack (see Section 4.3).

Notation. We denote the concatenation operator by $|$ and the XOR operator by $\oplus$.

3.2 Session Setup

As explained in Section 2, when an initiator A wants to communicate with a correspondent B, it first has to set up an *end-to-end session*. The goal of the session setup is 1) to test the *initiator* route (route between A and BS_A, containing a relays) and the *correspondent* route (route between BS_B and B, containing b relays), obtained from the underlying routing protocol, 2) to authenticate all nodes belonging to these routes, and 3) to inform these nodes about the traffic that will follow. A node can decide to not join the session, in which case the session setup fails and a new session is established using an alternative route. Successful completion of the session setup phase is a confirmation that both the initiator and correspondent routes are operational and that the end-to-end intermediate nodes accept to forward the traffic.

In order to set up a session, A generates an initiator session setup request message $AReq_0$ that contains a fresh request identifier $AReqID$ (e.g., generated in sequence), the initiator route $ARoute$, and some information $TrafficInfo$ about the traffic to be sent.[3] In addition, the request has a field $oldASID$ to carry the session ID of the broken initiator session, in case the request is sent to reestablish a broken session. This field is set to zero in the case of a new session establishment. Finally, $AReq_0$ contains a MAC computed by A using its secret key K_A:

$$AReq_0 = [AReqID \mid oldASID \mid ARoute \mid TrafficInfo \mid MAC_{K_A}(AReqID \mid oldASID \mid ARoute \mid TrafficInfo)].$$

Each forwarding node i $(1 \le i \le a)$ on the initiator route checks the traffic information $TrafficInfo$. If i decides to participate in the forwarding, then it computes a MAC on the whole message using its own key K_i, replaces the MAC in the request with the newly computed MAC, and forwards the request $AReq_i$ to the next hop (or to BS_A) where:

$$AReq_i = [AReqID \mid oldASID \mid ARoute \mid TrafficInfo \mid MAC_{K_i}(AReq_{i-1})].$$

Thus, when the request arrives to BS_A, it contains a single "layered" MAC that was computed by A and all the nodes on the initiator route in an iterative manner. BS_A then repeats all the MAC computations and checks the result against the MAC in the received request. It also verifies that the request ID is fresh (i.e., the message is not a duplicate) and, if the request is sent to reestablish a broken initiator session, it verifies that $oldASID$ corresponds to a valid session identifier previously initiated by A. If one of these verifications is not successful, then BS_A drops the request; otherwise, it sends the request, via the backbone, to the base station BS_B. BS_B generates and sends a correspondent session setup request $BReq_0$ toward B:

$$BReq_0 = [BReqID \mid oldBSID \mid BRoute \mid TrafficInfo],$$

where $BReqID$ is a fresh request identifier generated by the base station BS_B, $oldBSID$ is the session ID of the broken correspondent session, in case the request is sent to reestablish a broken session and $BRoute$ is the correspondent route.

Each forwarding node j $(1 \le j \le b)$ on the correspondent route computes and sends $BReq_j$ in the same way as the forwarding nodes in the initiator route did:

$$BReq_j = [BReqID \mid oldBSID \mid BRoute \mid TrafficInfo \mid MAC_{K_j}(BReq_{j-1})].$$

When B receives the request $BReq_b$, it returns to BS_B a correspondent session setup reply $BRep$ that contains the correspondent request ID $BReqID$ and a MAC that is computed over the received request $BReq_b$ (including the MAC therein) using the key K_B of B:

$$BRep = [BReqID \mid MAC_{K_B}(BReq_b)].$$

The reply is relayed back without any modifications to BS_B on the reverse route of the request. BS_B checks the "layered" MAC and, if it verifies correctly, BS_B informs BS_A that the session is valid. Then, BS_A (respectively, BS_B) sends an initiator (respectively, a correspondent) session setup confirmation message toward A (respectively B). The initiator session setup confirmation message $AConf$ contains the initiator request ID $AReqID$ and two freshly generated random numbers $AUSID$ and $ADSID$ representing the initiator session IDs to be used for packets sent from A to BS_A and from BS_A to A, respectively. It also contains a

3. The initiator A may not have any precise information about the traffic B will generate. $TrafficInfo$ is thus an estimate for the expected traffic in both directions. If A underestimates the traffic, the relaying nodes might interrupt the packet forwarding because the amount of data to forward is much larger than expected.

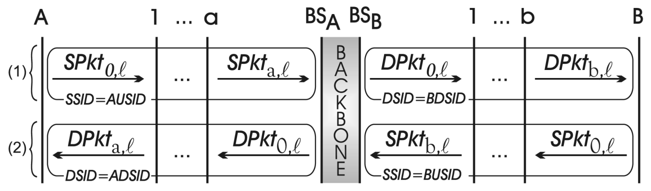


Fig. 2. The packet sending phase.

series of MACs where each MAC is intended for one of the nodes on the initiator route (including A):

$$\begin{aligned} AConf &= [AReqID \mid AUSID \mid ADSID \\ &\quad \mid AMAC_A \mid AMAC_1 \mid \ldots \mid AMAC_a], \\ AMAC_i &= MAC_{K_i}(AReqID \mid AUSID \mid ADSID \\ &\quad \mid oldASID \mid ARoute \mid TrafficInfo). \end{aligned}$$

The correspondent session setup confirmation $BConf$ has a similar structure:

$$\begin{aligned} BConf &= [BReqID \mid BUSID \mid BDSID \\ &\quad \mid BMAC_1 \mid \ldots \mid BMAC_b \mid BMAC_B], \\ BMAC_j &= MAC_{K_j}(BReqID \mid BUSID \mid BDSID \\ &\quad \mid oldBSID \mid BRoute \mid TrafficInfo). \end{aligned}$$

Each node on the initiator and correspondent routes (including A and B) verifies its own AMAC or BMAC and stores the two initiator or correspondent session IDs, respectively. The state information related to the established sessions (including session IDs, routes and cryptographic parameters) is stored in the operator's database. Then, using its secret key K_i and the session identifier, each node i involved in the communication generates a session key K'_i (e.g., $K'_i = h_{K_i}(SID)$, $SID = AUSID$ and $ADSID$ if i is in the initiator route, and $SID = BUSID$ and $BDSID$ if i is in the corresponding route, which leads to two session keys for each node, one for each direction of the communication) that it will use during the packet sending and the payment redemption phases. The base stations BS_A and BS_B also compute the session keys of all the nodes involved in the communication and save them locally.

The session becomes active for the base stations when they send the confirmation messages and for the nodes when they receive a valid confirmation message. Node i starts a timer t_i when it receives the request message; t_i is restarted each time i receives a valid message or packet that belongs to the session. Node i closes the session if t_i expires; closing a session means that the node discards all subsequent messages or packets that belong to the session. The nodes and the base stations keep state information in the memory until the acknowledgement and (if needed) packet receipts are sent to the operator (see Section 3.4).

Note that, in the case of initiator (respectively, correspondent) session reestablishment, it is not necessary to also reestablish the correspondent (respectively, the initiator) session if the latter is still valid. The broken session is reestablished using an alternative route and it is linked to the other (still valid) session in the operator's database.

3.3 Packet Sending

Once the session has been set up, S (which is A or B) starts sending packets to D.

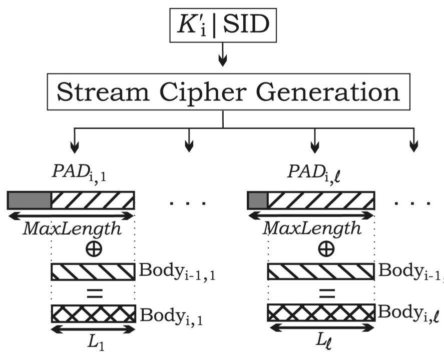


Fig. 3. Encryption of the packets.

The ℓth packet $SPkt_{0,\ell}$ sent by S contains the session ID $SSID$ (which is called $AUSID$ if $S = A$ and $BUSID$ if $S = B$), the sequence number ℓ, and the payload $Payload_\ell$. It also contains the "receipt seed" $SRcpt_{0,\ell}$ (details about the computation and the use of the receipts are given in Sections 3.4.1 and 3.4.4). In addition, S computes a MAC on the packet using the session key K'_S and encrypts the body of the packet (including the MAC) by XORing it with the pad $PAD_{S,\ell}$:

$$\begin{aligned} SPkt_{0,\ell} &= [SSID \mid SRcpt_{0,\ell} \mid \ell \mid Body_{0,\ell}] \\ \text{where} \quad SRcpt_{0,\ell} &= MAC_{K'_S}(SSID \mid \ell) \\ \text{and} \quad Body_{0,\ell} &= PAD_{S,\ell} \oplus [Payload_\ell \mid \\ &\qquad MAC_{K'_S}(SSID \mid \ell \mid Payload_\ell)]. \end{aligned}$$

The pads $PAD_{i,\ell}$ are generated by node i ($i = S$ for the source) as follows (see Fig. 3): The session ID $SSID$ ($DSID$ for the down-stream nodes) and K'_i are used as a seed to initialize the key stream generator of the stream cipher. Then, $PAD_{i,\ell}$ is chosen as the ℓth block of length $MaxLength$ of the generated key stream, where $MaxLength$ denotes the maximum allowed length of packets in bytes. If the length L_ℓ of the packet to be encrypted is smaller than $MaxLength$, then only the last L_ℓ bytes of $PAD_{i,\ell}$ are used, the rest of $PAD_{i,\ell}$ is thrown away.

The node i in the up-stream route (route between S and BS_S) verifies that the packet is not a duplicate, updates (and stores) the receipt[4] $SRcpt_{i,\ell}$ (details are in Section 3.4.4) and encrypts the body of the packet using the pad $PAD_{i,\ell}$:

$$\begin{aligned} SPkt_{i,\ell} &= [SSID \mid SRcpt_{i,\ell} \mid \ell \mid Body_{i,\ell}] \\ \text{where} \quad SRcpt_{i,\ell} &= MAC_{K'_i}(SSID \mid SRcpt_{i-1,\ell}) \\ \text{and} \quad Body_{i,\ell} &= PAD_{i,\ell} \oplus Body_{i-1,\ell}. \end{aligned}$$

When BS_S receives the packet, it retrieves the session keys of the nodes on the up-stream route, recomputes the pads, and removes all encryptions from the packet. If the resulting packet verifies correctly (i.e., it is not a duplicate and it has a valid MAC), the packet is forwarded[5] to the base station of the destination BS_D, otherwise it is dropped. BS_D changes the up-stream session ID to the corresponding

4. The receipt $SRcpt_{i,\ell}$ can be used by node i as a proof that it correctly received the packet $SPkt_{i,\ell}$ (see Section 3.4.1 for more details).

5. The packet is forwarded only if it is a data packet. The treatment of up-stream acknowledgement packets is presented in Section 3.4.2.

down-stream session ID $DSID$ (which is $BDSID$ if $S = A$ and $ADSID$ if $S = B$), computes a new MAC for D, computes the pad $PAD_{j,\ell}$ for each node j on the down-stream route (route between BS_D and D), including D, and encrypts the packet (including the MAC) by iteratively XORing it with all these pads. The result is:

$$\begin{aligned}&DPkt_{0,\ell} = [DSID \mid \ell \mid Body_{0,\ell}] \quad \text{where}\\ &Body_{0,\ell} = PAD_{1,\ell} \oplus \ldots \oplus PAD_{d,\ell} \oplus PAD_{D,\ell}\\ &\oplus [Payload_\ell \mid MAC_{K'_D}(DSID \mid \ell \mid Payload_\ell)].\end{aligned}$$

BS_D stores $MAC_{K'_D}(DSID \mid \ell \mid Payload_\ell)$ of every packet it sends together with the sequence number ℓ in order to be able to verify future destination acknowledgements and packet receipts. Note that, for the down stream, we do not need to add a field dedicated to the receipt; the receipt is generated using several fields of the down-stream packet (see Section 3.4.4).

Upon reception of $DPkt_{j-1,\ell}$, each node j computes and stores the receipt $DRcpt_{j,\ell}$ for the packet (as explained in Section 3.4.4), decrypts the body of $DPkt_{j-1,\ell}$ by XORing it with the pad $PAD_{j,\ell}$, and forwards the result $DPkt_{j,\ell}$ to the next hop where:

$$\begin{aligned}&DPkt_{j,\ell} = [DSID \mid \ell \mid Body_{j,\ell}] \quad \text{and}\\ &Body_{j,\ell} = PAD_{j,\ell} \oplus Body_{j-1,\ell}.\end{aligned}$$

When the packet reaches D, it removes the remaining encryption pad by XORing the packet with $PAD_{D,\ell}$. D can then verify the validity of the MAC generated by BS_D and store the MAC and ℓ for the generation of the acknowledgement (see Section 3.4.2). Note that for up-stream and down-stream packets, removing the encryptions and verifying the correctness of the resulting packet implicitly identifies the forwarding nodes and ensures that the packet took the right route.

3.4 Payment Redemption

3.4.1 Charging

As we have already mentioned in Section 2.2, charging and remuneration are performed by the network operator by manipulating the accounts of the nodes. When BS_S receives the packet Pkt_ℓ of length L_ℓ sent by the source S, the up-stream forwarding nodes are credited $\alpha(L_\ell)$ and the initiator A is charged $n(L_\ell)$. Both $\alpha(L_\ell)$ and $n(L_\ell)$ depend on the packet size and not on the number of forwarding nodes in the path. The operator will then take a loss for long routes but will make a profit from short routes. The charges and rewards should thus be set so that—relative to the average path length—the operator makes the desired profit.

The down-stream forwarding nodes are credited when Pkt_ℓ is acknowledged by D (see Section 3.4.2) because the operator may have no other reliable information about the delivery of the packet. The only incentive for D to not send the acknowledgement is to save resources. In order to discourage this misbehavior, D is charged a small amount ε when BS_D injects Pkt_ℓ in the down-stream route and is reimbursed when Pkt_ℓ is acknowledged. Note that, as the operator cannot distinguish between a packet loss and the case where D does not want to send the acknowledgment, it keeps the charge ε if no acknowledgement arrives for Pkt_ℓ.

If the packet is dropped or lost in the up-stream route, the nodes that relayed it can present the receipt for this packet (see Section 3.4.4) to the operator. The operator identifies the last node k ($1 \leq k \leq u$) in the path who sent a valid receipt for the packet and gives it a reward $\beta(L_{min})$, whereas the nodes that are before k in the path receive a reward $\alpha(L_{min})$, where L_{min} denotes the minimum length of a packet. This choice of reward is made because if the reward is higher than $\alpha(L_{min})$, the forwarding nodes may be tempted to drop short packets in order to get higher rewards than the ones they would get if they forward them. A is charged $n'(L_{min}) = (k-1) \cdot \alpha(L_{min}) + \beta(L_{min})$. Receiving $\beta(L_{min})$ can be perceived by k as its reward for informing the operator that the nodes 1 to $k-1$ in the path behaved properly. The β-reward should be sufficiently large to strongly counterbalance the cost c of forwarding the packet and the cost c' of maintaining and sending the receipt ($\beta \gg c$ and $\beta \gg c'$). The α-reward should also be substantially larger than β ($\alpha \gg \beta$) to prevent nodes from systematically dropping packets. Note that, even if c and c' are not constants (e.g., they depend on the battery level of the node), we can choose the α and β-reward in such a way that the conditions listed above are fulfilled.

If the packet is dropped or lost in the down-stream route, the nodes that relayed it are rewarded in a similar way as for the up-stream forwarding nodes, except for $\alpha(L_{min})$ and $\beta(L_{min})$ that are replaced by $\alpha(L_\ell)$ and $\beta(L_\ell)$, respectively, because the operator received the packet and knows its real length L_ℓ. The initiator A is fully charged $n(L_\ell)$.

3.4.2 Destination Acknowledgement

The destination D must acknowledge every packet it correctly receives. However, in order to save resources, it does not send acknowledgements on a per packet basis. Instead, the session is subdivided into "time periods" and the packets received during each period are acknowledged in a single batch. The acknowledgment $DAck_t$ of the tth time period of the session is formatted as the payload of a regular packet[6] and sent by D via the down-stream route to BS_D:

$$\begin{aligned}DAck_t =& [Batch_t \mid DFPkt_t \mid DLPkt_t \mid DLost_t \mid\\ & MAC_{K'_D}(Batch_t \mid DFPkt_t \mid DLPkt_t \mid DLost_t)],\end{aligned}$$

where $DFPkt_t$ and $DLPkt_t$ are the sequence numbers of, respectively, the first and the last received packets during the tth time period, $DLost_t$ is the list of the missing packets between $DFPkt_t$ and $DLPkt_t$ and

$$Batch_t = \bigoplus_{DFPkt_t \leq \ell \leq DLPkt_t;\ \ell \notin DLost_t} MAC_{K'_D}(DSID \mid \ell \mid Payload_\ell),$$

where $MAC_{K'_D}(DSID \mid \ell \mid Payload_\ell)$ is the MAC received in the packet Pkt_ℓ.

The packet is forwarded as a regular packet of the session. When BS_D receives it, the packet is decrypted and

6. It is necessary to be able to differentiate between a data packet and an acknowledgement (e.g., by using a flag bit).

identified as being an acknowledgement. Then, BS_D verifies the MAC and checks $Batch_t$ by XORing all the MACs of the packets from $DFPkt_t$ to $DLPkt_t$, excluding those in $DLost_t$ and comparing the result with the received value. If the verification fails, then BS_D ignores the acknowledgement. If BS_D does not receive $DAck_t$ during the $t+1\text{th}$ time period or if the throughput is not satisfactory (i.e., too many lost packets), an alternative route is used to establish a new session.

3.4.3 Up-Stream Acknowledgment

To attenuate the effect of several malicious attacks (see Section 4), the base station BS_S sends a single acknowledgment $UAck_t$ to S for all the packets it received during the $t\text{th}$ time period of the session. $UAck_t$ is sent in a regular packet and its format is similar to the format of $DAck_t$, except that the base station does not have to provide a $Batch$-like proof to the source:

$$UAck_t = [UFPkt_t \mid ULPkt_t \mid ULost_t \mid MAC_{K'_S}(UFPkt_t \mid ULPkt_t \mid ULost_t)].$$

When S receives $UAck_t$, it identifies it as being an acknowledgement and checks its validity by verifying its MAC. S can choose to reestablish the session to BS_S using an alternative route if no acknowledgement arrives for a given time period or if the throughput is unsatisfactory.

3.4.4 Packet Receipts

The concept of receipt we use in this paper is similar to the one used in [28]. It does not represent a proof that the node forwarded the packet but rather that it received it correctly. As we will see in Section 4.1, the use of the receipts helps to make packet forwarding rational.

For an up-stream forwarding node i, the receipt $SRcpt_{i,\ell}$ for the packet Pkt_ℓ is sent together with the payload and it is computed as explained in Section 3.3. We need a field dedicated to the receipt in the up-stream part of the communication because, if a part of the packet is used to compute the receipt, BS_S has no way to verify it in the case of packet loss, which is the very purpose of the receipts. For a down-stream forwarding node j, the receipt $DRcpt_{j,\ell}$ is computed as follows:

$$DRcpt_{j,\ell} = MAC_{K'_j}(DSID \mid M_{j,\ell}),$$

where $M_{j,\ell}$ represents the MAC field of the packet $DPkt_{j,\ell}$. It is possible for the operator to verify the receipts because it stores the MACs of the packets (they are also used to compute/verify the destination acknowledgements).

In order to save memory space, both up and down-stream forwarding nodes do not store the receipts for each packet but rather for a whole session; the forwarding node i stores a *batch* for each session it is involved in as a forwarding node:

$$Batch_{SID,i} = \bigoplus_{\ell \le LPkt\ ;\ell \notin Lost} Rcpt_{i,\ell},$$

where $LPkt$ is the sequence number of the last packet received so far and $Lost$ is the set of the sequence numbers of missing packets preceding $LPkt$.

Note that, for a node in the initiator route, $AUSID$ and $ADSID$ correspond to two distinct sessions. When a given session is closed and the last destination acknowledgement is sent, the operator informs the forwarding nodes, typically when the node is within the power range of a base station, about the rewards they received (e.g., using a packet similar to the up-stream acknowledgement). If a node i forwarded a packet Pkt_ℓ and was not paid for it, i sends the receipt to the operator. If the receipt is valid, the node is rewarded as explained in Section 3.4.1. A single receipt is sent to ask remuneration for several packets:

$$Rcpt_{SID,i} = [SID \mid Batch_{SID,i} \mid LPkt \mid Lost \mid MAC_{K'_i}(SID \mid Batch_{SID,i} \mid LPkt \mid Lost)].$$

Upon reception of this message, the operator verifies the MAC and if the verification is positive, it remunerates the node according to the rewarding scheme (see Section 3.4.1). Note that a node can ask for remuneration (by sending the receipt) even if it did not provide the service; this attack is studied in Section 4.1.

4 Security Analysis

In this section, we study the robustness of our set of protocols against the active attacks identified in Section 2.3.

4.1 Packet Dropping

In this attack, an attacker $\mathcal{M}$ that is part of the end-to-end route between S and D decides to drop a packet it is asked to forward. In this paragraph, we consider the effect of the attack on the different phases of our protocols and we show that this attack is not rational. This result proves, particularly for the packet sending phase, that our solution fosters cooperation.

Session setup phase. $\mathcal{M}$ can drop one or several of the following messages:

- The request message: The sender of the request (which is A or BS_B) does not receive the confirmation or the reply message, respectively. It then establishes a new session to the target (BS_A and B, respectively) using an alternative route. Note that dropping the request message is not necessarily an attack because the forwarding nodes can decide to not participate in a given session.
- The reply message: BS_B never receives the reply and the correspondent session setup fails. It then uses another route to establish the correspondent session.
- The confirmation message: Some of the nodes involved in the communication are not aware of the establishment of the session. If the initiator A is the source of the first packet to be sent during the session, we can have two cases: 1) $\mathcal{M}$ is in the initiator route, therefore A does not receive the confirmation message and considers that the session setup failed; it then establishes a new session using another route. 2) $\mathcal{M}$ is in the correspondent route, the session is then active for all the nodes, except for those that are after $\mathcal{M}$ in the correspondent route (including B); these nodes discard all the packets sent by A during the session. B is thus unable to send the periodic acknowledgment to BS_B and the session is reestablished. The problem is totally symmetric if B is the source of the first packet of

the session. In both cases, this attack is not rational and can be detected rapidly by the operator.

Packet sending phase. In this paragraph, we show that denying to forward packets is not rational; cooperation is thus the best choice for a selfish, rational node.

Proposition 1. *If a node i received a packet Pkt_ℓ to forward and if, later on, Pkt_ℓ was not acknowledged by the target (BS_S for the up-stream and D for the down-stream), then it is rational for i, once the session is closed, to send a receipt for Pkt_ℓ to the network operator.*

Proof. As explained in Section 3.4.4, after a given session is closed, the operator informs the nodes involved in that session about the rewards they received. If a node i correctly forwarded (or simply received) Pkt_ℓ and was not paid for it, i can send a receipt for it.

Sending a receipt $Rcpt$ of length L_{Rcpt} (see Section 5 for numerical values) represents a cost of $c'/NumPkts$ per packet, where $NumPkts$ denotes the number of packets received by i during the session and c' denotes the cost of sending $Rcpt$. Given the assumption of route stability (see Section 2.1), it is possible to neglect $c'/NumPkts$ in comparison with c (and, thus, in comparison with α and β) because $NumPkts$ is large.

If i decides not to send a receipt for Pkt_ℓ or if it sends an invalid receipt, then its payoff is:

- 0 if i dropped Pkt_ℓ during the packet sending phase,
- $-c$ if it forwarded Pkt_ℓ but none of the following nodes sent a valid receipt for it,
- $\alpha - c$ if it forwarded the packet and at least one of the following nodes in the path sent a valid receipt for the packet.

If i sends a valid receipt for Pkt_ℓ, then its payoff is:

- β if i dropped Pkt_ℓ during the packet sending phase,
- $\beta - c$ if it forwarded Pkt_ℓ but none of the following nodes sent a valid receipt for it,
- $\alpha - c$ if it forwarded the packet and at least one of the following nodes in the path sent a valid receipt for the packet.

Given that 1) a forwarding node cannot know if the receipt is valid or not before sending it to the operator, 2) the cost of sending the receipt is negligible, and 3) $\alpha \gg \beta \gg c$, we can state that sending the receipt is rational. □

Proposition 2. *If all the nodes involved in the communication are rational, then forwarding the packet Pkt_ℓ is rational for node i.*

Proof. As we will show in Section 4.3, the filtering attack is malicious. As the nodes involved in the communication are rational, they will not perform this attack on the packets they are asked to forward and, thus, the receipts produced by the intermediate nodes will be correct.

If node i decides to defect and drops a packet Pkt_ℓ it is asked to forward, i will still send a receipt for Pkt_ℓ since, according to Proposition 1, this is the rational behavior. The payoff of i would then be β.

If i decides to cooperate, then:

- If Pkt_ℓ reaches its target, then the payoff of i is $\alpha - c$.
- If, on the contrary, Pkt_ℓ does not reach its target, then at least one node j ($j > i$) will send a receipt for it (according to Proposition 1) and the payoff of i is also $\alpha - c$.

As we have $\alpha \gg \beta \gg c$, cooperation is rational for node i. □

Proposition 3. *If the route contains an attacker that repeatedly drops the packet Pkt_ℓ, then the network operator can identify it.*

Proof. As long as Pkt_ℓ is relayed by rational nodes, the packet is computed and correctly forwarded until it reaches the malicious node $\mathcal{M}$ that drops it. The rational nodes that are before $\mathcal{M}$ in the path will then send valid receipts for Pkt_ℓ (according to Proposition 1). The operator identifies the last node k in the path that sent a valid receipt, which is $\mathcal{M}$ or the rational node that is before it on the route (because $\mathcal{M}$ is also able to generate a valid receipt for the packet). The operator suspects then both k and $k+1$ of misbehavior. By crosschecking the information about different sessions and identifying the nodes that are suspected significantly more than average, the operator can identify the attacker and punish it in consequence. Note that, if $\mathcal{M}$ performed this attack only a few times, then the detection would be slower but the attack would be less harmful. □

Proposition 4. *Forwarding the packet Pkt_ℓ is rational for node i even if an attacker $\mathcal{M}$ will drop it later on.*

Proof. Node i has no information about whether the nodes after it in the path are rational or not. If it expects all of them to be rational, then the best choice for i is to cooperate (according to Proposition 2). If it expects node $i+1$ to be rational, then the best choice for i is to cooperate (its payoff would be $\alpha - c$ because according to Proposition 1, $i+1$ would send a receipt for the packet). Finally, if it expects node $i+1$ to be malicious and drop the packet, then the best choice for i is also to cooperate, because otherwise the operator would eventually believe it is malicious (according to Proposition 3) and would punish it. □

Payment redemption phase. The acknowledgement is encapsulated in a regular packet and the body is encrypted by all the nodes in the path, including the generator of the acknowledgement. An attacker $\mathcal{M}$ has thus no way to distinguish a packet containing an acknowledgement from a data packet, especially if some padding is used to prevent the acknowledgement packet from having a fixed and predefined length. A brute force attack would be for $\mathcal{M}$, in order to specifically drop the tth acknowledgement, to drop all the packets sent during the $t+1$th time period. The consequence of this attack is the reestablishment of the session using another route.

4.2 Replay Attack

We consider that a replay attack performed by an attacker $\mathcal{M}$ is successful if the replayed message or packet is considered as valid by *all* the parties involved in the communication (including the operator). Note that $\mathcal{M}$ is not necessarily part of the network. In this section, we will show that this attack is malicious and never successful.

Session setup phase. The operator maintains the information about all the sessions established so far. The replayed message (request, reply or confirmation) is thus detected by the first base station that receives it. A detection at the nodes is also possible; when a node i receives a replayed request message, it can identify it as a duplicate (and discard it) if:

- i is not part of the route in the request,
- i is supposed to be the initiator of the communication,
- or the session to be established is already active or it is closed but still in memory. Indeed, even if the mobile nodes do not keep track of all the messages and packets they received, they do maintain a short-term history (i.e., on-going sessions and session that are not acknowledged yet).

Packet sending phase. As for the session setup phase, the duplicate is detected by the first base station that receives it. But here, the intermediate nodes are also able to detect it because each forwarding node maintains the list of all packets it has received so far (for the computation of the receipt, see Section 3.4.4). The sequence number of the packet to forward corresponds then to the identifier of an already handled packet and the duplicate is discarded.

Payment redemption phase. The operator maintains the list of all acknowledgements and receipts it receives and can thus detect (and discard) a replayed message. Furthermore, as explained in Section 4.1, it is difficult to identify the packets containing the acknowledgements and, thus, to replay them specifically.

4.3 Filtering Attack

An attacker $\mathcal{M}$ that performs a filtering attack modifies one or several fields of the packet it is asked to forward. In this section, we analyze the effect of this attack on our protocols. We also consider the *free-riding* attack where two colluders $\mathcal{M}_1$ and $\mathcal{M}_2$, on the end-to-end route, attempt to piggyback data (using appending or substitution) on the exchanged packets, with the goal of not having to pay for the communication.

Session setup phase. $\mathcal{M}$ can tamper with:

- The request or the reply messages: The verification of the "layered" MAC fails and the base station (BS_A or BS_B) discards the message. A new session is then established using an alternative route.
- The confirmation message: The first node that receives the tampered message discards it because the verification of the MAC fails. If $\mathcal{M}$ tampers with one (or more) MAC(s) in the message, the first node whose MAC was modified and that receives the message discards it. This attack has the same effect as dropping the confirmation message (see Section 4.1) and is detected in the same way.

The fields of the session setup messages are not encrypted. It is then possible for two colluders $\mathcal{M}_1$ and $\mathcal{M}_2$ to piggyback information. However, the size of fields is small enough to make the sending of useful data very long and fastidious.

Packet sending phase. $\mathcal{M}$ can tamper with the different fields of the packet Pkt_ℓ.

- Modifying SID, ℓ or $Body_{i,\ell}$ is detected by the target of the packet (BS_S for the up-stream and D for the down-stream) because the "layered" MAC does not verify correctly.
- We hereafter define the *early duplicate* attack, a malicious attack where $\mathcal{M}$ creates a fake packet with a sequence number ℓ that it expects to be used by the legitimate source in the (near) future. This packet is considered as valid by the intermediate nodes (because they cannot verify it) but it is discarded at the target because the MAC is not correct. However, when the source sends the "real" ℓth packet, the forwarding nodes consider it as a duplicate and, thus, discard it. Our protocols, as presented so far, are vulnerable to this attack. If the operator wants to attenuate the effect of this subtle attack, it can do so (at the cost of a small overhead) by making use of hash chains (i.e., a chain of N hash values where w_N is chosen at random, $w_{N-i} = h(w_{N-i+1}), 0 < i \leq N$, and h is a one-way hash function).

Let us first describe the solution for the initiator session. During the session setup phase, the base station BS_A sends the first hash values AUw_0 and ADw_0 of two sufficiently long hash chains, in the initiator confirmation message, to the nodes in the initiator route (including A). BS_A also sends the hash value AUw_m encrypted with the secret key of A in the confirmation. A can thus retrieve the elements 0 to m of the hash chain and send the hash value AUw_ℓ ($1 \leq \ell \leq m$) with the ℓth packet it generates.[7] BS_A sends the hash value ADw_ℓ with the ℓth packet it sends toward A. The intermediate nodes can verify the validity of the hash values by checking that $w_0 = h^\ell(w_\ell)$ ($w = AUw$ or ADw). The verification of the hash value can be optimized if we use mechanisms such as [8] for example. The packets containing invalid hash values are discarded.

The solution is totally symmetric for the correspondent session. Note here that, given w_ℓ, one can retrieve the hash values of all the previous packets in the sesssion. This means that packet out of order should be discarded. But, this constraint is logical in our case because we use the notion of sessions. All the packets are then expected to go through the same route and to arrive in order; the contrary is thus suspicious.

The use of the hash values can also solve the case where the attacker tampers only with w_ℓ; the attack is detected at the first node that receives the modified packet because the checking of the hash value fails.

Modifying both w_ℓ and ℓ is an even more subtle malicious attack. Let us assume that a forwarding node receives the packets $Pkt_{\ell-1}$ and Pkt_ℓ to forward. It discards $Pkt_{\ell-1}$ and replaces the sequence number and the hash value in Pkt_ℓ by $\ell - 1$ and $w_{\ell-1}$, respectively. The sequence number and the hash

7. When A is about to run out of hash values, the base station provides it (in the same way the up-stream acknowledgment is sent) with a hash value AUw_{m+n}. A can then compute n new valid hash values.

value are considered as valid by the following forwarding nodes. Of course, the packet is discarded at the target because the MAC is not correct. The attack is possible if the attacker is part of the route and, thus, all the nodes on the route are suspected by the operator. The first direct effect of this attack is for the source to cancel the session because the throughput is too low; the second effect is that the operator eventually, by crosschecking the information about the suspected nodes, identifies the attacker.

- The free-riding attack is not rational during the packet sending phase; the data sent by $\mathcal{M}_1$ cannot be interpreted by $\mathcal{M}_2$ because it was encrypted at least b one intermediate node.[8] If this attack is performed anyway, it is detected as a "regular" filtering or packet dropping attack (depending on whether $\mathcal{M}_2$ forwarded the tampered packet or not).
- Modifying only the receipt $SRcpt$ in the up-stream packets (there is no field dedicated to receipts in the down-stream packets) is a malicious attack. If the base station BS_S detects such an attack (the packet is correct but the receipt is not), then it reestablishes the session (if $S = B$) or asks the initiator to do it (if $S = A$). Such a radical solution is needed because, as explained in Section 3.4.4, the nodes maintain one batch per session by XORing all the receipts of the packets they handled. If one of these receipts is incorrect, then the batch is incorrect and the receipt does not verify correctly at the operator.
- The attacker $\mathcal{M}$ can tamper with the packet it is asked to forward but without altering the fields used by the intermediate nodes to generate the receipts. The following nodes in the route forward the modified packet. When the target (BS_S or D) receives it, it detects the attack and reestablishes the session.

Payment redemption phase. This attack is similar to the packet dropping attack during the payment redemption phase.

4.4 Emulation Attack

This attack is equivalent to the cloning of a SIM card in a GSM cellular network and can be detected in the same way; a node claiming to be in several physical locations simultaneously (e.g., it is in two geographically distinct cells) is automatically suspected by the operator. Furthermore, statistical methods can be used to determine whether certain nodes relay more traffic than is reasonable, given the type of the node. Either of these events suggests that the device is dishonest.

4.5 Hybrid Attacks

So far, we have analyzed the effect of each of the four active attacks we identify in Section 2.3. However, more sophisticated attacks can combine two or more of the attacks described so far. For example, two colluders $\mathcal{M}_1$ and $\mathcal{M}_2$ that are on the same route may want to perform, respectively, the filtering attack and the packet dropping attack. If the filtering attack does not modify the information needed by the intermediate nodes to compute the receipts, the operator will detect a "regular" packet dropping attack and will identify $\mathcal{M}_2$ as being the attacker (see the proof of Proposition 3). If, on the contrary, the nodes that are between $\mathcal{M}_1$ and $\mathcal{M}_2$ are not able to generate valid receipts, then $\mathcal{M}_1$ will be identified by the operator as an attacker that performed a filtering attack (see the appendix, which can be found on the Computer Society Digital Library at http://computer.org/tmc/archives.htm). The same reasoning can be applied to the case where there are more than two colluders.

8. Having two colluding nodes that are neighbors and that perform the free-riding attack makes no sense because they can communicate directly with each other.

4.6 Securing the Routing Protocol

As stated in Section 2.4, even if the underlying routing protocol is not secure, the operator is able to detect several routing attacks. Indeed, during the session setup, the initiator and correspondent routes are tested and the nodes belonging to these routes are authenticated, which allows the operator to detect attacks such as routing loops or invalid routes. However, some routing attacks cannot be detected before the packet sending phase (e.g., *Gratuitous detour*, *Black hole*, or *Gray hole* attacks [13]); the network operator can then employ statistical methods to detect them. Note that securing the routing protocol is out of the scope of this paper; we therefore consider, to exemplify, the following attacks that we believe are the most pertinent regarding our solution:

***Gratuitous detour* attack**. In this attack, the adversary makes the route appear longer by adding virtual nodes [13]. The operator determines statistically if the set of intermediate nodes is inconsistent (e.g., an emulated node is in the route or an attacker is performing the wormhole attack) or if the route is much too long (a route in hybrid ad hoc networks is not expected to be long, having a too long routes is therefore suspicious). The operator can also suspect such an attack if two or more nodes seem to be always neighbors, despite mobility. More heuristics can be found in [15].

***Black or gray holes* attack**. This attack is similar to the packet dropping attack during the packet sending phase.

5 Overhead

In this section, we estimate the communication and computation overheads of the solution we have described. Reasonable values of the size of the different fields appearing in our protocol are provided in Table 1. *NbFwdrs* is the number of forwarding nodes on the route (up-stream or down-stream), ℓ is the sequence number of the packet, and *NbLostPkts* is the number of packets lost during the session or the time period.

The request ID and the session IDs are encoded on 4 bytes each to reduce the risk of using the same identifier for two different requests or sessions. The field *Route* is the concatenation of the 16 byte identifiers (assuming, e.g., an IPv6 format) of the nodes. The *TrafficInfo* field is used to inform the forwarding nodes about the traffic to be generated; using 16 bytes to encode it seems to be reasonable. Finally, we encode ℓ on 2 bytes to support long sessions and *SRcpt* on only 1 byte because its computation and storage should be lightweight.

5.1 Communication Overhead

Session Setup Phase. According to Table 1, establishing an end-to-end session with $NbFwdrs$ forwarding nodes (in

TABLE 1
Size of the Fields Used in Our Protocol (for Both Up and Down-Streams)

Field Name	ReqID	SID	Route	TrafficInfo	MAC	ℓ	SRcpt	Lost
Size (bytes)	4	4	*NbFwdrs**16	16	16	2	1	*NbLostPkts**2

TABLE 2
Simulation Results for the Different Values of the Speed (Pause Time = 0 s)

AvrSpeed	5.6 m/s			6.7 m/s			7.8 m/s		
	NbFwdrs	AvrLT (s)	95% CI	NbFwdrs	AvrLT (s)	95% CI	NbFwdrs	AvrLT (s)	95% CI
	1.3	10.7	2.1	1.4	8.1	1.7	1.5	7.8	1.5

each of the routes) represents an overhead of $156 + NbFwdrs * 64$ bytes.

The session setup overhead is directly related to the lifetime of the sessions, which, in turn, very much depends on the stability of the routes.

Description of the simulations. We consider a network composed of 100 nodes laid out on a $500 \times 500\text{m}^2$ single cell and one base station situated in the center of the cell. We fix the power range of the nodes and the base station to 100 m. We use the random waypoint mobility model [16] with a 0 s pause time and we discard the first 1,000 seconds of simulation time to remove the initial transient phase [7]. We perform three sets of simulations where the speed is uniformly chosen between x and 10 m/s, $x = 2$, 3, and 4 m/s [26], which corresponds to an average speed *AvrSpeed*= 5.6, 6.7, and 7.8, respectively; we run 100 simulations for each value of *AvrSpeed*. As we are interested in the lifetime of the routes and not in communication interface, our silulation is written in plain C++ instead of ns.

Figures of interest. In our simulations, we are interested in the two following figures:

- The average lifetime of a route (*AvrLT*): After the initial transient phase of each simulation, we randomly choose a node that has a route to the local base station (we choose the shortest path, the effect of mobility on the performance of more sophisticated routing protocols is discussed in [2]) and we observe the lifetime of this route. The simulation ends when at least one link on the route is broken. *AvrLT* represents the average value of all these lifetime values over the 100 simulations.
- The average number of forwarding nodes (*NbFwdrs*): This number is computed for the node we consider for the *AvrLT*.

Results. The results, given in Table 2, show that the stability of the routes decreases with higher mobility of forwarding nodes. For *AvrLT*, we consider a 95 percent confidence interval (*CI*).

In order to estimate the amount of information that a node can send during this period of time, let us consider the case where the nodes are running a *Voice over IP* application using a G.711 Codec (Rate = 64 kbit/s) with a frame size (including the headers) of 200 bytes [10]. If we consider that the average speed = 7.8 m/s, the route remains stable for an average of 7.8 s; it is possible during this period to send 62.4 kbytes of data. The overhead of an end-to-end session setup is 252 bytes (the average number of forwarding nodes is 1.5), which represents only 0.4 percent of the amount of information (payload) that is possible to send during the session. Moreover, as explained in Section 3.2, it is possible to reestablish only the broken session (the initiator session or the correspondent session), which reduces this overhead.

The presence of one (or more) active malicious attackers in the end-to-end route can also lead to a session reestablishment. However, the operator can statistically identify the attacker(s) (see Section 4); the risk of being identified and punished represents a disincentive to cheat.

Packet Sending Phase. Considering the field sizes of Table 1, we can see that the packet sending phase represents an overhead of 23 bytes for up-stream packets and 22 bytes for down-stream packets. If the packet size is 200 bytes (considering again the VoIP example), the overhead represents at most 11.5 percent of the packet size. This overhead is reduced if we use larger packets.

Sending the Acknowledgment. The destination acknowledgement and the up-stream acknowledgement are generated each time period and their sizes are $36 + 2 * NbLostPkts_t$ bytes and $20 + 2 * NbLostPkts_t$ bytes, respectively. The receipt $Rcpt_{SID,i}$ is a $23 + 2 * NbLostPkts$ bytes message that the node i sends directly (i.e., without relaying) to the operator once per session. We expect the number of packets lost to be small in both cases (i.e., acknowledgement and receipt); otherwise, the session is reestablished because the throughput is not satisfactory.

5.2 Computation Overhead

In this section, we consider the computation overhead for the mobile nodes. The overhead is expressed in terms of battery consumption and number of computations. However, as shown in [23], we can consider the battery consumption, due to cryptographic computations, as negligible compared to the energy needed for data transmission.

Session Setup Phase. This operation requires all the nodes to perform 1 MAC computation and 1 MAC verification each.

Packet Sending Phase. For each packet, the source and the destination have to perform one MAC operation each. However, the main overhead in this phase is represented by the usage of stream cipher encryption (performed by the source and all the forwarders), which ensures the authentication of the nodes involved in the communication and prevents the free-riding attack. But, stream ciphers are very fast, and some operate at a speed comparable to that of 32 bit CRC computation [12].

Acknowledgment computation. For the destination acknowledgement, D performs one MAC computation/time period and one XOR operation/packet. For the up-stream acknowledgement, S performs one MAC verification/time period. Finally, for the receipts, each forwarding node performs one MAC computation/time period and one XOR operation/packet.

Numerical example. As an example, a Celeron 850 MHz processor under Windows 2000 SP can perform a MAC computation (and verification) with HMAC/MD5 algorithm at 99.863 Mbytes/s and a stream cipher encryption (and decryption) using the Panama Cipher (little endian) algorithm at 120.301 Mbytes/s [12]. These numbers provide an order of magnitude; if slower (or faster) processors are used, they would of course scale correspondingly.

6 Related Work

In this section, we discuss some research efforts related to the issues of the cooperation of nodes in (pure) ad hoc networks and in hybrid ad hoc networks.

Cooperation in ad hoc networks. Several research groups have considered the problem of selfishness and the stimulation of cooperation in mobile ad hoc networks. In [9], Félegyházi et al. establish the connection between the ad hoc network topology and the possible existence of cooperation. In [20], Marti et al. consider the case where a node agrees to cooperate but fails to do so. Their solution uses a "watchdog" mechanism to identify the misbehaving nodes and a "pathrater" mechanism to construct routes that avoid those nodes. Both the CONFIDANT [5] and the CORE [22] approaches propose a reputation based solution to identify and punish misbehaving nodes. In [28], Zhong et al. rely on a central authority that collects receipts from the forwarding nodes and charges/rewards the nodes based on these receipts. In [6], Buttyán and Hubaux use a virtual currency (nuglets) to charge/reward the packet forwarding service provision in ad hoc networks.

Cooperation in hybrid ad hoc networks. In [17], Lamparter et al. propose a rewarding scheme to encourage cooperation in hybrid networks (i.e., mobile ad hoc networks with access to the Internet, which they call "stub ad hoc networks"). They assume the existence of an Internet Service Provider that authenticates the nodes involved in a given communication and takes care of charging or rewarding them. However, [17] and our current approach present two main differences. First of all, in [17], the authors analyze the robustness of their solution only against rational attacks, whereas, in our proposal, we consider malicious attacks as well. The second difference is that the cryptographic functions used in [17] are based on public-key cryptography, whereas our solution is based solely on symmetric key cryptography, which is more suitable for resource constrained mobile devices.

In [15], we have proposed a micropayment scheme for hybrid ad hoc networks that encourages collaboration in packet forwarding. However, our current proposal significantly differs from [15] in many aspects. First of all, in [15], we assume an asymmetric communication model, where the up-stream communication is potentially multihop and the down-stream communication is *always* single-hop, whereas, in this paper, both the up-stream and the down-stream communications are potentially multihop. Second, in [15], the nodes report a fraction of their packet forwarding actions (on a probabilistic basis) to an accounting center that consequently remunerates the nodes. The approach we propose here does not rely on reports; instead, we use the concept of session during which each forwarding node authenticates itself to the base station by altering the packet to be forwarded in a specific way. Finally, the protocol proposed in [15] includes routing decisions, whereas the protocols that we propose in this paper are independent of routing.

7 Conclusion

In this paper, we proposed a set of protocols that fosters cooperation for the packet forwarding service in hybrid ad hoc networks. Our solution is based on the charging and rewarding of the nodes and relies exclusively on symmetric cryptography to comply with the limited resources of most mobile stations. We have used the concept of sessions, which takes advantage of the relative stability of routes, and we have shown that our scheme stimulates cooperation in hybrid ad hoc networks. Finally, we have analyzed the robustness of our protocols against various attacks and have shown that our solution thwarts rational attacks and detects malicious attacks.

As future work, we intend to consider techniques that aim at the calibration of the relevant parameters, and to study the reaction of the network to sophisticated attacks (e.g., by means of simulations). We will also explore further the statistical detection, at the operator, of malicious attacks and we will study the coexistence of several operators.

Acknowledgments

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information an Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

[1] G.N. Aggélou and R. Tafazolli, "On the Relaying Capacity of Next-Generation GSM Cellular Networks," *IEEE Personal Comm.*, Feb. 2001.

[2] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A Framework to Systematically Analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks," *Proc. INFOCOM Conf.*, 2003.

[3] Y. Bejerano, "Efficient Integration of Multi-Hop Wireless and Wired Networks with QoS Constraints," *Proc. Mobicom Conf.*, 2002.

[4] N. Ben Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson, "A Charging and Rewarding Scheme for Packet Forwarding in Multihop Cellular Networks," *Proc. MobiHOC Conf.*, 2003.

[5] S. Buchegger and J.-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes—Fairness in Distributed Ad Hoc NeTworks," *Proc. MobiHOC Conf.*, 2002.
[6] L. Buttyán and J.-P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, 2003.
[7] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Comm. and Mobile Computing*, vol. 2, no. 5, 2002.
[8] D. Coppersmith and M. Jakobsson, "Almost Optimal Hash Sequence Traversal," *Proc. Conf. Financial Cryptography*, 2002.
[9] M. Félegyházi, J.-P. Hubaux, and L. Buttyán, "Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks," to appear.
[10] B. Goode, "Voice Over Internet Protocol (VoIP)," *Proc. IEEE*, vol. 90, Sept. 2002.
[11] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications.* Wiley, 2002.
[12] http://www.eskimo.com/~weidai/benchmarks.html, year?
[13] Y.-C. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Proc. Mobicom Conf.*, 2002.
[14] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterli, "Towards Self-Organizing Mobile Ad-Hoc Networks: the Terminodes Project," *IEEE Comm. Magazine*, vol. 39, no. 1, pp. 118-124, Jan. 2001.
[15] M. Jakobsson, J.-P. Hubaux, and L. Buttyán, "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," *Proc. Conf. Financial Cryptography*, 2003.
[16] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, 1996.
[17] B. Lamparter, K. Paul, and D. Westhoff, "Charging Support for Ad Hoc Stub Networks," *J. Computer Comm.*, Summer 2003.
[18] Y.-D. Lin and Y.-C. Hsu, "Multihop Cellular: A New Architecture for Wireless Communications," *Proc. INFOCOM Conf.*, 2000.
[19] O.C. Mantel, N. Scully, and A. Mawira, "Radio Aspects of Hybrid Wireless Ad Hoc Networks," *Proc. Vehicular Technology Conf.*, 2001.
[20] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. Mobicom Conf.*, 2000.
[21] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.
[22] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation In Mobile AD HOC Networks," *Proc. Conf. Comm. Multimedia Security*, 2002.
[23] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. Mobicom Conf.*, 2001.
[24] V.W. Kipp, "The Battle of NIMBY," http://www.findarticles.com/p/articles/mi_m0LEF/is_2002_August_1/ai_91033662, 2002.
[25] H. Wu, C. Qios, S. De, and O. Tonguz, "Integrated Cellular and Ad Hoc Relaying Systems: iCAR," *IEEE J. Selected Areas in Comm.*, vol. 19, no. 10, Oct. 2001.
[26] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," *Proc. INFOCOM Conf.* , 2003.
[27] A.N. Zadeh, B. Jabbari, R. Pickholtz, and B. Vojcic, "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)," *IEEE Comm, Magazine*, June 2002.
[28] S. Zhong, Y.R. Yang, and J. Chen, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks," *Proc. INFOCOM Conf.*, 2003.

Naouel Ben Salem received the Engineering diploma in computer science from ENSI (**E**cole **N**ationale des **S**ciences de l'**I**nformatique, Tunis), Tunisia in 2000. As a research assistant with the Laboratory for Computer communication and Application (LCA), she is now working toward the PhD degree in communication systems at EPFL. Her research interests include self-organization (especially in ad hoc networks), cooperation in wireless networks, and network security. For more information, check http://lcawww.epfl.ch/salem. She is a student member of the IEEE.

Levente Buttyán received the MSc degree in computer science from the Budapest University of Technology and Economics (BUTE) in 1995 and the PhD degree from the Swiss Federal Institute of Technology—Lausanne (EPFL) in 2002. In 2003, he joined the Department of Telecommunications at BUTE, where he currently holds a position as an assistant professor and works in the Laboratory of Cryptography and Systems Security (CrySyS). His research interests are in the design and analysis of security protocols for wired and wireless networks with a special emphasis on wireless sensor networks and ad hoc networks. More information can be found at http://www.crysys.hu/.

Jean-Pierre Hubaux joined the faculty of EPFL in 1990; he was promoted to full professor in 1996. His research activity is focused on mobile networking and computing, with a special interest in fully self-organized wireless ad hoc networks. In particular, he has performed research on cooperation aspects, security, power efficiency, and distributed algorithms for ad hoc and sensor networks. During the last few years, he has been strongly involved in the definition and launching phases of a new National Competence Center in Research named "Mobile Information and Communication Systems" (NCCR/MICS), see http://www.terminodes.org. Within his first year at EPFL, he defined the first curriculum in communication systems. From October 1999 until September 2001, he was the first chairman of the Communication Systems Department. He is an associate editor of the *IEEE Transactions on Mobile Computing* and of the *Elsevier Journal on Ad Hoc Networks*. He also served as the general chair for the Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), held on the EPFL campus. He has been serving on the program committees of numerous conferences and workshops, including Infocom, Mobicom, and Mobihoc. He has held visiting positions at the IBM T.J. Watson Research Center and at the University of California at Berkeley. Previous to that, he spent 10 years in France with Alcatel, where he was involved in R&D activities, mostly in the area of switching systems architecture and software. For more information, please check http://www.lcawww.epfl.ch/hubaux. He is a senior member of the IEEE.

Markus Jakobsson received the PhD degree in computer science from the University of California at San Diego in 1997. He held a joint appointment at the San Diego Supercomputer Center and General Atomics during 1996 and 1997, and joined Bell Laboratories as a member of the technical staff in 1997. In 2001, he joined RSA Laboratories as a principal research scientist, where he stayed until joining Indiana University at Bloomington in 2004. He has been an adjunct associate professor at New York University. He is the inventor or coinventor of over 50 patents. He is the vice president of the International Financial Cryptography Association. Dr. Jakobsson is pursuing research relating to cybersecurity and cryptographic protocols. Among other topics, he is interested in cryptographic techniques suitable for use in low-power wireless environments; privacy issues, in particular as they relate to payments, voting, Web browsing and routing; payment schemes and the use of economic mechanisms as incentives for collaboration; and techniques for making electronic commerce secure and reliable. He is also interested in social engineering and phishing and the prevention of these attacks.